VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NAVIGAČNÍ SYSTÉM DO BUDOV NA MOBILNOM ZARIADENÍ

NAVIGATION SYSTEM FOR BUILDINGS ON MOBILE DEVICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETER ŽIŠKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÍTĚZSLAV BERAN, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Žiška Peter**
Obor: Informační technologie
Téma: **Navigační systém do budov na mobilním zařízení**
Navigation System for Buildings on Mobile Device
Kategorie: Uživatelská rozhraní

Pokyny:

1. Prostudujte možnosti lokalizace mobilního zařízení v budovách, vytváření map a jejich využití pro úlohy navigace v budově a plánování nejkratší cesty v mapě.
2. Navrhněte a implementujte systém, který umožní uživateli vybrat cíl v budově (osoba, místnost), na základě odhadu pozice uživatele v budově naplánuje cestu k cíli a s využitím dat z inerciální jednotky bude uživatele navigovat k cíli.
3. Vytvořte postup a vhodné podpůrné nástroje pro efektivní vytvoření nové mapy budovy.
4. Vytvořte mapu FIT a demonstrujte vlastnosti řešení. Detekujte slabá místa řešení a diskutujte možnosti dalšího vývoje.
5. Vytvořte plakát a krátké video prezentující klíčové výsledky vašeho řešení.

Literatura:

- L. Mathis. *Designed for Use: Create Usable Interfaces for Applications and the Web*, Pragmatic Bookshelf, ISBN-10: 1934356751, 2011
- L. Shklar, R. Rosen. *Web Application Architecture: Principles, Protocols and Practices*, Wiley, ISBN-10: 047051860X, 2009
- Dále dle pokynu vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a částečně body 2 a 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Beran Vítězslav, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Práca je zameraná na návrh a vytvorenie postupu pre jednoduchšie nasadenie navigačného systému do budov. Výsledné riešenie sa skladá z dvoch častí: pomocného nástroja pre tvorbu lokalizačných a navigačných podkladov a mobilnej navigačnej aplikácie. Pomocný nástroj vytvorí podklady potrebné pre lokalizovanie pomocou skenovania okolitých access pointov na mieste státia užívateľa v mapovanej budove. Nástroj tiež vygeneruje graf pre navigovanie so spoluprácou užívatela a na základe geometrie miestností v budove. Lokalizácia mobilného zariadenia je založená na metóde WIFI fingerprinting, ktorá v prvej časti potrebuje mať k dispozícií mapu fingerprintov v budove, takže nevyhnutným krokom je takúto mapu manuálne vytvoriť. Nástroj uľahčí užívateľovi vytvoriť potrebnú mapu a zefektívni proces nasadenia navigácie do budov na mobilné zariadenia.

Abstract

This work is focused to suggest an approach, which simplifies creation of indoor navigation systems on mobile devices. The final solution is divided into two parts: an auxiliary tool for creating localization and navigation assets and mobile navigation application. The auxiliary tool creates assets needed to localize with the help of scanning surrounding access points in the place of user's standing in a building. The tool also generates navigational graph with cooperation of user and based on the geometric representation of building's rooms. The localization of a mobile device is based on WIFI fingerprinting method, which in the offline phase needs to have a map of fingerprints available. It is necessary to create this map manually. The tool will make easier for the user to create needed map and make deployment of the indoor navigation into the mobile devices more efficient.

Klíčová slova

Lokalizácia, navigácia, android, wifi fingeprinting, generovanie navigačného grafu, zber fingerprintov

Keywords

Localization, navigation, android, wifi fingeprinting, navigational graph generation, collection of fingerprints

Citace

ŽIŠKA, Peter. *Navigační systém do budov na mobilnom zariadení*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

Navigační systém do budov na mobilnom zariadení

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vítězslava Berana, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Žiška
12. května 2018

Poděkování

Rád by som sa poďakoval Ing. Vítězslavovi Beranovi, Ph.D za jeho pomoc a poskytovanie užitočných rád, ktoré mi otvárali nové možnosti pri riešení tejto bakalárskej práce.

Obsah

Kapitola 1

Úvod

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať mobilnú aplikáciu, ktorá vyhľadá pozíciu užívateľa a naviguje ho k cieľu vo vnútornom prostredí budovy, ktorý si užívateľ vyhľadá. Táto aplikácia by teda mala umožňovať zobrazenie mapy a pozície užívateľa a taktiež vizuálne zobrazenie cesty pri prípadnom navigovaní k cieľu. Ďalej mojou úlohou bolo navrhnúť pomocné nástroje, ktoré umožnia vytvoriť novú mapu a nakoniec demonštrovať vytvorené postupy pri vytvorení mapy budovy Fakulty informačných technológií v Brne.

V prvých kapitolách som rozobral teoretické základy, ktoré sú potrebné pre lepšie pochopenie implementovaného postupu a tvorby. V prvom rade sú spomenuté lokalizačné techniky, ktoré je možné využiť v priestoroch budov, pretože GPS nie je možné použiť kvôli zlej priepustnosti signálu cez steny. Ďalej som sa venoval teórií vedúcej k samotnému navigovaniu a to konkrétne reprezentácií navigácie a možnostiach ako takúto reprezentáciu vytvoriť.

Pomocou teoretických základov som v kapitole 4 navrhol spôsob, akým je možno reprezentovať materiály potrebné pre lokalizovanie a navigovanie, ktoré je možné využiť v implementácií navigačného systému do budov na mobilnej aplikácií. V tejto kapitole som navrhol spôsob, ktorým možno získať dáta potrebné pre navigovanie pomocou metódy WIFI fingeprinting, kde je nutné mať vytvorenú databázu fingerprintov. Ďalej je navrhnutý spôsob vytvorenia navigácia pomocou grafu viditelnosti za prítomnosti geometrie jednotlivých miestností.

V kapitolách 5 a 6 som uviedol detaily implementácie nástroja, ktorý využíva navrhnuté postupy a mobilnej aplikácie, ktorá využíva výstupy z implementovaného nástroja. Na záver týchto kapitôl sú popísené testy, ktoré boli vykonané pri testovacej fáze vývinu mobilnej aplikácia a pomocného nástroja a to, hlavne testovanie lokalizácie, ktorá je dôležitou súčasťou každého navigačného systému, taktiež som získal dáta, ktorými je možné definovať vplyv času na vytvorenú databázu fingerprintov.

V záverečnej kapitole som uviedol zhodnotenie celého postupu a implementácie jednotlivých častí systému. Uvažoval som o budúcnosti tohto postupu a hlavne ako by bolo možné vylepšiť postup tvorby databázy pre fingerprinty a samotné generovanie grafu.

Kapitola 2

Lokalizácia

Určenie polohy osoby alebo zariadenia v priestore je základnou časťou, ktorú musí obsahovať každá dobrá navigácia. Najznámejším a najpresnejším spôsobom lokalizácie je Global Positioning System (GPS), ktorý však pre úlohy lokalizovania v budovách nie je vhodný a to pre nedostatočnú silu signálu, ktorá nedokáže prejsť cez steny, ktorá je hrubšia ako strecha auta. Ale čo ak nastane situácie, kedy sa potrebujete zorientovať vo vnútri budovy? Pre túto situáciu existuje veľke množstvo prístupov, ktoré tento problém riešia. Metódy môžu byť založené či už na Wi-fi infraštruktúre, ktorá je oporným bodom tejto práce, Bluetooth technológiách alebo s využitím kamery na mobilnom zariadení.

V nasledujúcich riadkoch sa kladie dôraz práve na tieto techniky. Ďalej budú zhodnotené vlastnosti vybraných techník a v neposlednom rade opísaná funkcionalita.

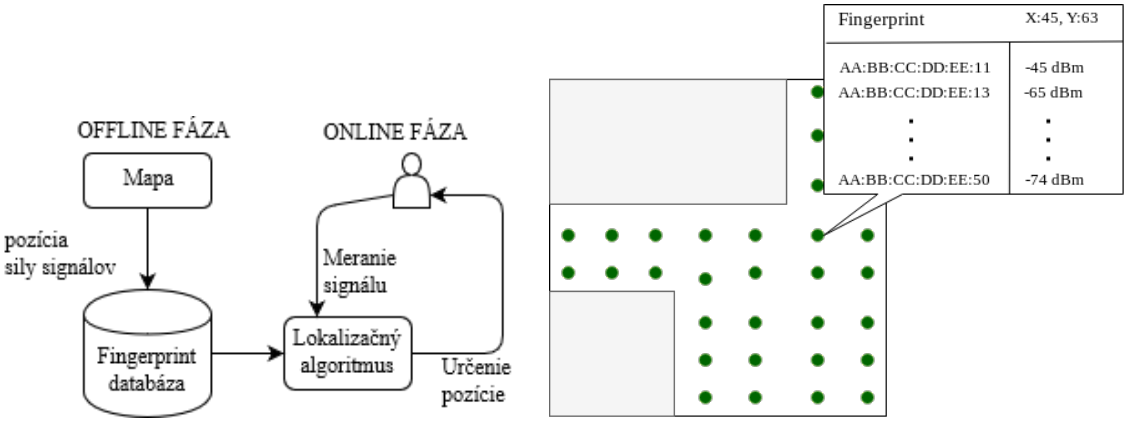
2.1 Wifi Fingerprinting

Metóda fingerprinting je zložená z dvoch fáz: z offline fáze – prvá fáza a z online fáze – druha fáza. Priebeh metódy je znázornený na obrázku č. 2.1a. V offline fáze sa vytvára mapa bodov, kde bod reprezentuje vektor síl signálov dostupných prístupových bodov. Tieto vektory sú následne uložené v databáze, ktorá bude slúžiť ako referenčná porovnávacia pomôcka pre určenie polohy [9]. Mapa fingerprintov je zobrazená na obrázku 2.1b.

V online fáze, užívatel, ktorý chce určiť svoju polohu, napríklad pomocou mobilného zariadenia, sníma aktuálne signály, kde získa taktiež vektor signálov. Tento vektor je ďalej porovnaný s databázou vytvorenou v offline fáze. Porovnanie môže prebiehať pomocou intervalu, ktorý sa zvolí na základe síl signálov z online fáze a ďalej sa hľadajú sily signálov z databáze, ktoré zodpovedajú zvolenému intervalu. Záznam, ktorý má najväčší počet síl signálov v zodpovedajúcom intervale je prehlásený ako pozícia užívateľa.

Ďalšou možnosťou porovnania fingerprintov je použitie metriky. Najčastejšie sa využíva Euklidovská vzdialenosť medzi referenčným bodom z užívateľského zariadenia a záznamami z databáze. Vyberie sa tá, ktorá ma najmenšiu Euklidovskú vzdialenosť a prehlási sa ako pozícia užívateľa. Pre výpočet tejto vzdialenosti sa využíva vzorec 2.1, kde d_i je vzdialenosť fingerprintu od aktuálnej polohy, |A| je počet prístupových bodov, R a R^* sú hodnoty signálov rovnakých prístupových bodov. Hodnoty signálov sú merané v jednotkách decibel-miliwatoch [dBm].

$$d_i = \sqrt{\sum_{k=0}^{|A|}(R_k - R_k^*)^2} \tag{2.1}$$



(a) Priebeh metódy Fingerprinting. Offline fáza značí zber fingerprintov do databáza. Tá sa zase využíva v online fáze na porovnávanie s dátami, ktoré získa užívateľ z jeho zariadenia, pričom sa tento proces porovnania pravidelne opakuj. Obrázok je dosptupný online[1].

(b) Vytvorená databáza fingeprintov v priestore, kde bodka značí fingerprint udávajúci vektor síl signálov s korešpondujúcimi MAC adresami dostupných access pointov na danej pozícií.

Obrázek 2.1: Metóda WIFI fingerprinting jej časti a reprezentácia mapy fingerprintov.

Presnosť ovplyvňujú rôzne faktory a to napríklad počet skenov na priestor a teda čím väčšia zrnitosť vzorkov, tým je metóda presnejšia. Prenosnosť dokáže ovplyvniť samozrejme aj počet access pointov v okolí. V rôznych literatúrach je metóda presná na 3–12 metrov. Ale ako každá metóda aj táto má výhody a nevýhody, ktoré sú ďalej vymenované.

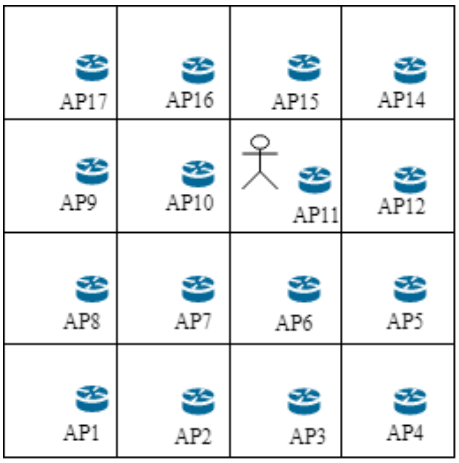
- Výhody
  - Umožňuje využiť už existujúcu infraštruktúru, ktorú nie je potreba modifikovať
  - Veľmi dobrá presnosť
  - Možnosť použiť vo vnútri budov ale aj vo vonkajších priestoroch
- Nevýhody
  - Pri veľkej komplexnej budove, je potreba mať veľkú databázu, s ktorou budeme porovnávať
  - Vytvorenie databázy je časovo zložitý proces

2.2 Cell of origin

Cell of origin [1] je jednou z najjednoduchších techník získania približnej pozície. Jej princíp spočíva vo vytvorení databáze jednotlivých častí priestoru, ktorým priradíme identifikačné číslo a vysielač, ktorý má najsilnejšiu silu signálu na danom mieste. Rozdelenie priestoru môžte vidieť na obrázku č. 2.2. Pri zistovaní pozície zariadenia v tomto priestore, je potrebné získať silu signálu vysielačov. Vysielač s najsilnejším signálom použijeme na prehľadanie v databáze a to tak, že v databáze nájdeme tento vysielač a vrátime časť priestoru, ktorá zodpovedá vybranému vysielaču z databáze.

[1] https://www.semanticscholar.org/paper/Wi-Fi-Fingerprint-Based-Indoor-Positioning%3A-Recent-He-Chan/3e7aabaffdb4b05e701c544451dce55ad96e9401?tab=abstract

- Výhody
  - Využitie už existujúcej infraštruktúry
  - Možnosť použiť vo vnútri budov ale aj vo vonkajších priestoroch
- Nevýhody
  - Nedostatočne presná



Obrázek 2.2: Metóda Cell of origin, ktorá rozdeľuje priestor do buniek a pre jednotlivé bunky vyberie ten access point, ktorý má v danej bunke najväčší signál. Na obrázku možno vidieť užívateľa, ktorého približná poloha je v úrovni AP11.

2.3 Trilaterácia

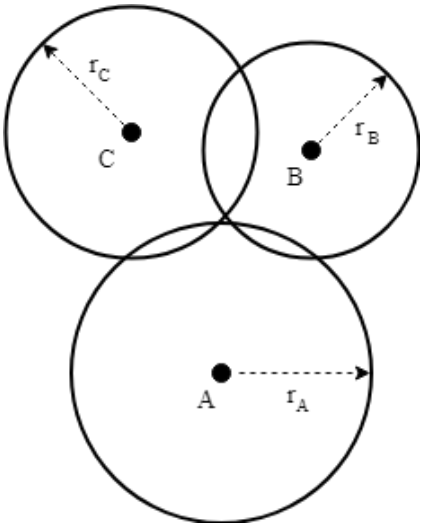
Trilaterácia [5] je proces, ktorým určujeme absolutnú alebo relatívnu polohu pomocou geometrie kružníc. Podmienkou, aby táto technika pracovala správne, je počet vysielačov a to konkrétne tri .

Zmyslom téjto metódy je výpočet vzdialenosti z vysielača k prijímaču. Táto vzdialenosť vytvorí pomyselnú kružnicu, kde by sa dané zariadenie mohlo nachádzať. K zníženiu možností sa použijú ďalšie vypočítané vzdialenosti z vysielačov, ktoré sú v dosahu. Takto získané vzdialenosti nám vytvoria kružnice, kde priesečník určuje pozíciu prijímača (zariadenia). Na obrázku č. 2.3 je zobrazený jej koncept. Ďalej sú uvedené metódy založené na trilaterácií: Time of Arrival, Time Difference of Arrival a Received Signal Strength.

Time of Arrival (ToA)

Time of Arrival [12] meria čas, ktorý potrebuje signál k príchodu ku zariadeniu a naspäť k vysielaču. Potrebné je mať zosynchorizované hodiny vo vysielači a prijímači a samozrejme aj minimálne tri vysielače ako je spomenuté pri trilaterácií.

- Výhody



Obrázek 2.3: Princíp trilaterácie, body A, B, Cpozície vysielačov. Sily signálov respektíve polomery kružníc sú dané pomocou r_A, r_B, r_C. Priesečník kružníc určuje polohu lokalizovaného zariadenia.

  - Implementácia pre otvorené priestory

- Nevýhody
  - Minimálne tri zariadenia v dosahu
  - Pozícia vysielačov musí byť presne daná
  - Zlá spolupráca s WiFi

Time Difference of Arrival (TDoA)

Time Difference of Arrival [4] je podobná metóda ako ToA s tým rozdielom, že vzdialenosť je vypočítaná z rozdielu časov, ktoré sú namerané z vysielača k prijímaču a z prijímača k vysielaču. Táto metóda zdieľa nevýhody s ToA.

Received Signal Strength

Tento prístup trilaterácie je založený na sile signálu z prístupových bodov, ktorých poloha je vopred presne známa. Tieto sily signálov vytvoria pomyselnú kružnice, kde hľadáme priesečníky aspoň troch týchto kružníc. Súradnice, ktoré získame priradíme polohe zariadeniu, ktoré prijíma tieto sily signálov. Táto metóda vynikajúco funguje vo vonkajšom svete, kde nie je veľa prekážok, ktoré môžu signál znížiť. Tento problém nastáva v budovách, kedy steny odrážajú alebo znižujú signál a tak dochádza ku skresleným silám signálov, ktoré zariadenie príjme.

- Výhody
  - Podporuje WiFi
  - Pozícia zariadenia musí byť presne daná

  - Dobrá voľba pre vonkajšie priestory

- Nevýhody
  - Je potrebné mať minimálne 3 zariadenia v dosahu
  - Pozícia zariadenia musí byť presne daná
  - Vo vnútorných priestoroch dosahuje nepresné výsledkyýsledky

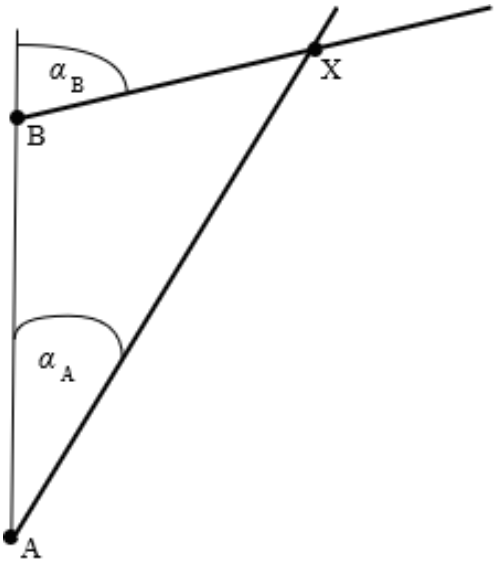
2.4 Triangulácia

Angle of Arrival

Metóda [13] určuje pozíciu na základe uhlov príchodu signálu z vysielačov na prijímač. Pre získanie uhlov dopadu signálu je potrebný špeciálny vysielač so smerovou anténou. Geometrickými vzťahmi medzi uhlami dokážeme získať bod, kde sa priamky s daným uhlom pretínajú. Získaný bod reprezentuje pozíciu prijímača. Pre lokalizáciu v 2D priestore stačí mať dva vysielače s fixnou pozíciou. V prípade 3D priestoru ich je potreba mať minimálne tri.

- Výhody
  - Podporuje real-time získanie polohy
  - Vnútorné a vonkajšie lokalizovanie

- Nevýhody
  - Potrebné špeciálne antény

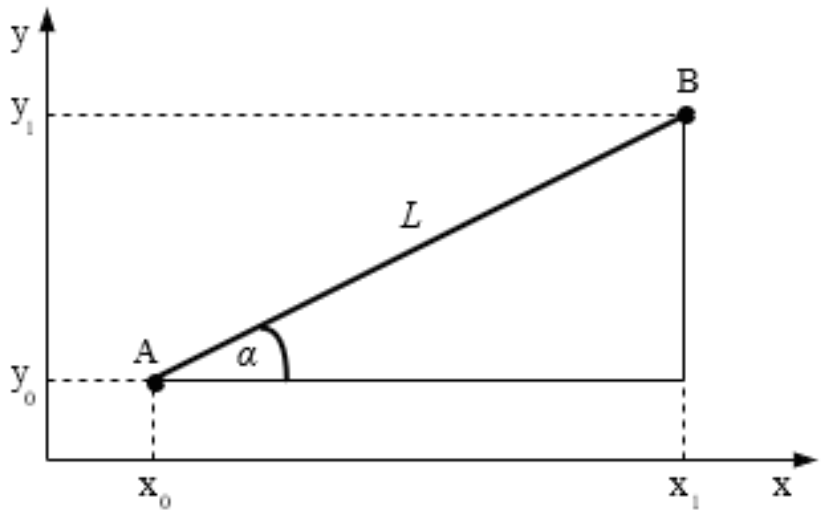


Obrázek 2.4: Triangulácia. Body `A` a `B` sú počiatočné body, ktoré zvierajú uhol α_B respektíve α_A kedy pretnutia týchto uhlov určujú lokalizovanú pozíciu `X`.

2.5 Dead Reckoning

Dead Reckoning [11] (DR) je výpočet pozície zariadenia na základe známej počiatočnej pozície a na vzdialenosti získanej zo senzorov v mobilnom zariadení. Vzdialenosť je vypočítaná pomocou akcelerometru, ktorým detekujeme kroky užívateľa a tak aj rýchlosť, ktorou sa pohybuje. Ďalším krokom je získanie smeru pohybu. Toto umožňuje poskytnúť mobilné zariadenia, ktoré obsahuje zabudovaný kompas. Ten určí smer, ktorým sa zariadenie pohybovalo. DR sa v minulosti využívalo pre navigáciu ponoriek [8]. S pribúdajúcimi zariadeniami, ktoré obsahujú akcelerometer, kompas a rôzne iné senzorové zariadenia sa s touto metódou začalo počítať aj v úlohách lokalizácie zariadení v budovách. Obrázok 2.5 zobrazuje princíp metódy Dead Reckoning.

- Výhody
  - Využitie akcelerometra
  - Stačí mobilné zariadenia
- Nevýhody
  - Je potrebné poznať štartovnú pozíciu
  - Problematická detekcia zmeny poschodí



Obrázek 2.5: Počiatočnú štartovnú pozíciu reprezentuje usporiadaná dvojica $[x_0,y_0]$, uhol α udáva smer, ktorým sa zariadenie pohybovalo. L je vzdialenosť, ktorú zariadenie prešlo. Výsledná poloha je daná usporiadanou dvojicou súradníc $[x_1,y_1]$.

2.6 Systémy využívajúce kameru

Zaujímavé riešenie pre získanie polohy užívateľa prinášajú mobilné telefóny so zabudovanou kamerou. Tieto metódy využívajú rozpoznanie obrazu poprípade špeciálne značky, ktoré nesú informáciu o polohe. Ďalej budú tieto metódy podrobnejšie popísené.

Vizuálne značky

Vizuálna značka je typ dvojdimenzionálneho čiarového kódu. Do tohto kódu môžu byť zakódované informácie URL, prihláseniu sa do webovej služby alebo niesť informácie o cene produktu, na ktorom sa nachádza. Najčastejšie je využitie URL a to aj pri určovaní polohy zariadenia. Poznáme rôzne typy značiek, ako napríklad Quick Response Code (QR), ARToolKit Marker alebo Aztec Code, ktoré príklady zobrazenia sú uvedené na obrázkoch 2.6, 2.7 a 2.8.

Pri určovaní polohy sú do značky zakódované informácie o polohe poprípade ďalšie informácie, ktoré chce poskytovateľ ponúknuť užívateľovi. Následne sú nakonfigurované značky poumiestňované po budove. Po zosnímaní značky pomocou mobilného zariadenia je zobrazená informácia čitateľná pre človeka.

Táto technika sa zvyčajne využíva v múzeách alebo rôznych priestoroch s expozíciami, pretože tieto značky môžu niesť rôzne dodatočné informácie ako napríklad odkaz na video alebo hlasový záznam, ktoré si následne užívateľ môže spustiť a tak možno simulovať interaktívneho sprievodcu.

- Výhody
  - Nízke náklady na vybudovanie infraštruktúry so značkami
  - Stačí mobilné zariadenia
- Nevýhody
  - Neumožňuje real–time lokalizáciu

Obrázek 2.6: QR Code

Obrázek 2.7: ARToolKit Marker

Obrázek 2.8: Aztec Code

Rozpoznanie snímkov

Metóda [10] patrí do kategórie analýzy scén. Scéna je v tomto prípade chápaná ako snímok vnútra budovy. Porovnávanie obrazu z kamery s databázou obrazov scény má najväčšie využitie v scénach s úzkymi chodbami a v scénach s malým počtom uhľov pohľadu.

Delí sa na dve časti: offline a online fázu. V offline fáze sa vytvára databáza obrazov za pomoci video sekvencie nadobudnutej z miestnosti. Tieto dáta sú uložené aj s pevne

danými informáciami o pozícií pre každú snímku zo sekvencie. V online fáze sa porovnáva každá snímka s nadobudnutou snímkou z kamery. Najpresnejšia zhodná snímka z databáze sa vyberie a záznam o pozícií asociovaný s týmto snímkom sa prehlási za pozíciu zariadenia.

- Výhody
  - Umožňuje určiť orientáciu
- Nevýhody
  - Časová náročnosť vytvorenia databázy snímkov
  - Náročný výpočet pri veľkej databáze

Kapitola 3

Navigácia

V tejto kapitole je rozobraná teória zaoberajúca sa navigáciou, reprezentáciou navigačnej mapy a algoritmami riešiacimi nájdenie najkratšej cesty v grafe.

3.1 Princípy navigácie

Navigácia môže byť popísaná ako cesta, ktorá vychádza z počiatočného bodu do cieľového bodu, zatiaľ čo vieme očakávať, čo nás na tejto ceste bude čakať. Táto cesta by mala byť pochopiteľná pomocou vizuálnych alebo sluchových vnemov. Najčastejším spôsobom zobrazenia navigácia je zobrazenie mapy na elektronickéj obrazovke a po zadaní cieľového bodu sa zobrazí trasa, ktorú by mal užívateľ nasledovať, aby dosiahol cieľ svojej cesty.

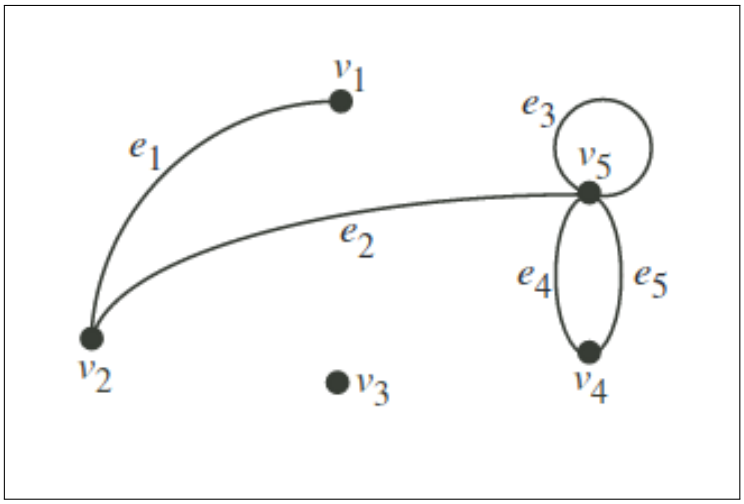
Nevyhnutná informácia pre navigačný systém je počiatočný a cieľový bod. Počiatočný bod je zvyčajne definovaný pomocou lokalizačných techník ako napríklad tie, ktorým bola venovaná pozornosť v kapitole č.2 pre lokalizovanie vo vnútorných priestoroch. Cieľový bod môže užívateľ zadať pomocou koordinátov, vizuálne kliknutím na mapu poprípade vyhľadaním v zozname miest, ktoré môžu byť navštívené. Cesta, ktorá ma byť reprezentovaná počiatočným a koncovým bodom by nemala byť náhodná, ale najlepšie tá najkratšia, ktorá užívateľovi ušetrí čas alebo v navigácií v automobile aj peniaze. Ďalej bude táto problematika najkratšej cesty zanalyzovaná v samostatnej podsekcií.

3.2 Reprezentácia navigácie

Pre vytvorenie navigačného systému je nutné mať mapu, v ktorej bude možno prehľadávať priestor čo najjednoduchším spôsobom. Naskytuje sa jednoduché riešenie, využitie grafov s ohodnotenými hranami, kde sa pomocou algoritmov na to určených, dokáže vyhľadať najkratšia cesta.

Konceptuálne je graf [15, 14] tvorený uzlami a hranami medzi jednotlivými uzlami. Definícia grafu z teórie grafov hovorí: majme graf G tvorený dvojicou (V, E), kde $V = \{v_1, \ldots, v_n\}$ je neprázdna množina vrcholov a $E \subseteq V \mathrm{x} V$ je množina usporiadaných dvojíc vrcholov z množiny V a nazývame ju množina hrán. Hrany vždy začínajú a končia v uzle, väčšinou sú vrcholy odlišné ale môžu byť aj rovnaké a tak tvoria slučku. Príklad definovania grafu a jeho zobrazenia je popísaný na obrázku 3.1. Na obrázku je možné taktiež vidieť aj slučku, ktorá je označená hranou e a vedie z vrcholu v_5 do rovnakého vrcholu a to v_5. Zobrazený graf je neorientovaný, čiže hrany medzi vrcholmi sú obojsmerné a tak je možné ísť z jednoho uzlu do druhého a naspäť.

Druhým typom sú orientované grafy. Orientácia je zobrazená pomocou šípiek, ktoré predstavujú smer, ktorým je možné prejsť z počiatočného uzlu do konečného uzlu. Príkladom z reálneho sveta môže byť sieť ciest, kde môžu existovať aj jednosmerné cesty.



Obrázek 3.1: Zobrazenie a definícia neorientovaného grafu. Graf $G = (V, E)$, kde je množina uzlov $V = \{v_1, v_2, v_3, v_4, v_5\}$ a množina hrán $E = e_1, e_2, e_3, e_4, e_5$

.

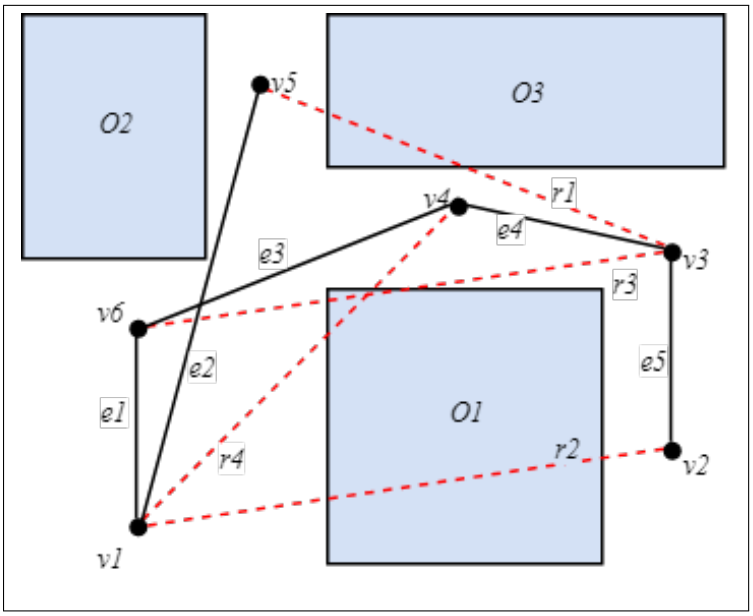
Graf viditeľnosti

Reprezentácia grafovej štruktúry sa zaoberá objektami, ktoré môžu tvoriť prirodzenú prekážku medzi dvoma vrcholmi. Tieto prekážky sú definované pomocou polygonionálnej reprezentácie. Preto pre výpočet grafu viditelnosti [2] je nutné nájsť všetky dvojice, medzi ktorými sa daná prekážka nenachádza. Vypočet týchto dvojíc prebieha pomocou testovania úsečiek medzi jednotlivými vrcholmi nato, že sa medzi nimi nenachádza žiadna prekážka. Ak sa žiadna prekážka nenachádza v ceste, táto hrana sa vytvorí a uloží do grafovej štruktúry, inak sa hrana do grafovej štruktúry neuloží. Na obrázku č. 3.2, je možno vidieť zobrazenie takto vytvoreného grafu viditeľnosti. Graph obsahuje vrcholy `v1, v2, v3, v4, v5 a v6`, medzi ktorými figurujú hrany `e1, e2, e3, e4 a e5`, pričom sú tu zobrazené aj niektoré hrany (červená farba) `r1, r2, r3, a r4`, ktoré sú v grafe viditeľnosti odstránené, pretože im v ceste bráni prekážka.

Zložitosť testu, ktorým zisťujeme, či sa medzi dvoma vrcholmi nachádza prekážka je $O(n)$ kde n je počet vrcholov, ktoré sa v grafe nachádzajú. Graf viditeľnosti sa využíva v robotike, kde sa hľadá taká cesta, ktorá neprechádza cez žiadne kolízie – inými slovami, vo vybranej ceste nenarazí robot na žiadnu prekážku.

Bresenhamov algoritmus

Pre testovanie či sa medzi vrcholmi grafu nenachádza prekážka môže využiť Bresenhamov algoritmus. Tento algoritmus síce uvažuje o vykreslovaní medzi dvoma bodmi, ale modifikáciou získame požadované vlastnosti.



Obrázek 3.2: Ukážka zobrazenia grafu viditeľnosti. Na obrázku je možno vidieť tri prekážky s označením `O1, O2` a `O3`a vrcholy `v1, v2, v3, v4, v5 a v6`, medzi ktorými figurujú hrany `e1, e2, e3, e4 a e5`.

Algoritmus [7] pracuje s celými číslami a využíva pre výpočet len operácie sčítania, odčítania a porovnania. Pre tieto vlastnosti je tento algoritmus jeden z najefektívnejších a najpoužívanejších v oblasti vykreslovania úsečky v resterizovanom priestore.

Úsečka sa vykresluje pixel po pixly od počiatočného bodu $P_1 = [x_1, y_1]$ až po cieľový bod $P_2 = [x_2, y_2]$. V x-ovej osy postupujeme tak, že zvyšujeme túto súradnicu o $d_x = 1$ takže platí $x_{i+1} = x_i + d_x$. Posun v y-ovej súradnice vypočítame pomocou znamienka prediktoru p. Výpočet prediktoru predpokladá smernicový tvar rovnice úsečky uvedený v rovnici č. 3.1. Vypočítame si vzdialenosti d_1 a d_2 vzťahom uvedeným rovnicami 3.2 a 3.3. Vyjadríme si rozdiel hodnôt $\Delta d = d_1 - d_2$, ktorého znamienko sa využije pre rozhodnutie o ďalšej hodnote y_{i+1}. Nato aby sme sa zbavili realnej hodnoty k, vynásobíme vzťah pre Δd s hodnotu Δx ($k = \frac{\Delta y}{\Delta x}$)(rovnica 3.4).

$$y = kx + q, \qquad k = \frac{\Delta x}{\Delta y} \tag{3.1}$$

$$d_1 = y_{i+1} - y_i = k(x_i + 1) + q - y_i \tag{3.2}$$

$$d2 = y_i + 1 - y_{i+1} = y_i + 1 - k(x_i + 1) - q \tag{3.3}$$

$$p_i = \Delta d \Delta x = 2\Delta y x_i - 2\Delta x y_i + 2\Delta y + \Delta x(2q - 1) \tag{3.4}$$

Ak je hodnota prediktoru $p_i < 0$, to znamená, že vzdialenosť $d_2 > d_1$, potom pixel na y-ovej ose leží v aktuálnej hodnote y_i a platí, že $y_{i+1} = y_i$. V druhom prípade je hodnota y-vej osy navýšená o $d_y = 1$, teda platí $y_{i+1} = y_i + d_y$.

3.3 Vyhľadanie najkratšiej cesty

Problém vyhľadania najkratšej cesty je typickým problémom teórie grafov. Typicky sa pomocou nich optimalizujú rôzne problémy v dopravných alebo komunikačných sietach. Na-

príklad nájdenie najratšej cesty z miesta A do miesta B alebo z informatického hladiska nájdenie najkratšej cesty pre paket z siete A do siete B.

Ako ohodnotenia hrán sa zvyčajne využívajú kladné prirodzené nezáporné čísla. Pri potrebe ohodnotenia hrany záporným číslom je dôležité myslieť nato aby v takomto grafe nevznikol záporný cyklus (súčet ohodnotení hrán tohto cyklu je záporný). V tomto prípade sa neoplatí hľadať najkratšiu cestu, pretože vždy je možné nájsť kratšiu cestu, ktorá vedie cez záporný cyklus. Pretože môžeme rôzne veľa krát prejsť cez cyklus a tak sa najkratšia cesta vždy zníži. Pre graf so záporným ohodnotením hrán je najvhodnejšie využit Bellman–Fordov algoritmus.

Pre vyhľadávanie najkratšej cesty existujú rôzne algoritmy ako už spomenutý Bellman–Fordov algoritmus ale aj veľmi známy Dijkstrov algoritmus poprípade A*. Tieto algoritmy budú bližšie popísené v nasledujúcich riadkoch.

Dijkstrov algoritmus

Dijkstrov algoritmus [3] je algoritmus, ktorý hľadá najmenej ohodnotenú cestu z počiatočného bodu až do bodu cieľového. Ďôležitým predpokladom využiteľnosti tohto algoritmu je, aby ohodnotenie hrán malo priradené nezáporné hodnoty.

Algoritmus si uchováva pre každý uzol hodnotu najkratšej cesty, ktorá do neho vedie. V počiatočnom stave prehľadávania má každý vrchol hodnotu najkratšej cesty rovnú $d = \infty$. Výnimku tvorí počiatočný uzol, ktorého hodnota je $d = 0$. Na začiatku algoritmus nájde susedné uzly, ktoré sú definované v grafovej reprezentácií grafu ako hrany s jednoznačným ohodnotením. Následne je susedným uzľom vypočítaná nová najkratšia vzdialenosť d podľa ohodnotených hrán a uložia sa do množiny nenavštívených uzlov N. Ďalšia etapa pozostáva z pridania počiatočného uzlu do množiny navštívených uzľov P a pridelenia predchodcu, z ktorého je určená najkratšia vzdialenosť. Výberom uzlu s najkratšou cestou z množiny N, ktorý bude v ďalšej iterácií chápaný ako počiatočný, sa tento postup opakuje až dokým sa nevyprázdni množina N alebo ak sa hľadá konkrétny cieľ, tak sa vyhľadávanie ukončí vtedy, keď sa tento cieľový uzoľ vyberie z množiny N. Pre lepšiu predstavu fungovania tohto algoritmu je uvedený pseudokód algoritmu č. 1.

Dijkstrov algoritmus je využiteľný aj v neohodnotenom grafe, kedy sú všetky hrany umiestnené v grafe ohodnotené ako $d = 1$.

Časová zložitosť samotného algoritmu je kvadratické $O(n^2)$, kde n je počet vrcholov grafu, v ktorom sa najkratšia cesta vyhľadáva. Samozrejme algoritmus môže skončiť skôr v prípade dvoch blízkych vrcholov grafu. Zložitosť algoritmu môže byť kubická $O(n^3)$ v prípade, že chceme nájsť dĺžky najkratších ciest medzi všetkými dvojicami vrcholov.

Algoritmus A*

Pre vyhľadávanie najkratšej cesty sa využíva aj algoritmus A* [6], ktorý hľadá najkratšiu cestu v neorientovanom grafe. Tento algoritmus využíva heuristické funkciu, ktorá definuje ďalší smer smerovania algoritmu až k cieľu. Heuristická funkcia vypočíta odhad pre každý vrchol v grafe a podľa tejto odhadnutej hodnoty sa rozhoduje, do ktorého ďalšieho vrcholu budú kroky algoritmu nasledovať. Asi najpoužívanejšou heuristickou funkciou je výpočet Euklidovskej vzdialenosti od vrcholu až po požadovaný cieľový vrchol grafu.

Nevýhodou tohto prístupu je to, že algoritmus hľadá najpodobnejšiu cestu k priamke z počiatku až po cieľ. Toto má za následok ignorovanie najkratšej cesty, ktorá sa v grafe

Algoritmus 1: Dijkstrov Algoritmus

Input : $Graph = (Vertex, Edge), source$
Output: X_t

```
1:  distance[source] = 0
2:  foreach Vertex V in Graph do
3:      if (V ≠ Source) then
4:          distance[v] = ∞
5:      Q.add(V)
6:  end foreach
7:  while (!Q.isEmpty()) do
8:      v = Q.getMinimumDistance()
9:      Q.remove(v)
10:     foreach neighbor u of v do
11:         alt = distance[v] + length(v, e)
12:         if (alt < dist[u]) then
13:             distance[u] = alt
14:     end foreach
15: end while
16: return distance[]
```

nachádza. Ďalšou nevýhodou je to, že pri neexistencií cesty sa prehľadajú všetky uzly grafu, čo môže trvať pomerne dlhý čas. Naopak výhodou je jednoduchosť implementácie.

Časová zložitosť algoritmu je závislá na výbere optimálnej heuristickej funkcie. V najhoršom prípade je počet vrcholov exponencionalný v porovnaní s počtom vrcholov najkratšej cesty. Algoritmus A* sa využíva v umelej inteligencií pre navigovanie vo veľkom pristore, kde existujú prekážky, ale je použiteľný aj pre najdenie najkratšej cesty danej grafom.

Bellman–Fordov algoritmus

Pri grafoch, ktoré majú hrany ohodnotené pomocou záporných hodnôt je Dijkstrov algoritmus nevyužiteľný, preto vznikol Bellman–Fordov algoritmus [6], ktorý v takto ohodnotenom grafe dokáže nájsť najkratšiu cestu. Z praktického hľadiska sa tento algoritmus využíva v smerovacom protokole RIP (Routing information protocol), ktorý slúži pre smerovanie dát v sieti. Najkratšia cesta je nájdená pomocou najmenšieho počtu hrán medzi počiatočným vrcholom a cieľovým vrcholom grafu.

Zložitosť sa udáva ako $O(n * m)$, kde n je počet vrcholov a m je počet hrán grafu. Vykonaný výpočet je však pomalší ako to je u Dijkstrovho algoritmu. Preto sa využíva najmä tam, kde sa v ohodnoteniach hrán môže objaviť záporná hodnota.

Kapitola 4

Návrh riešenia

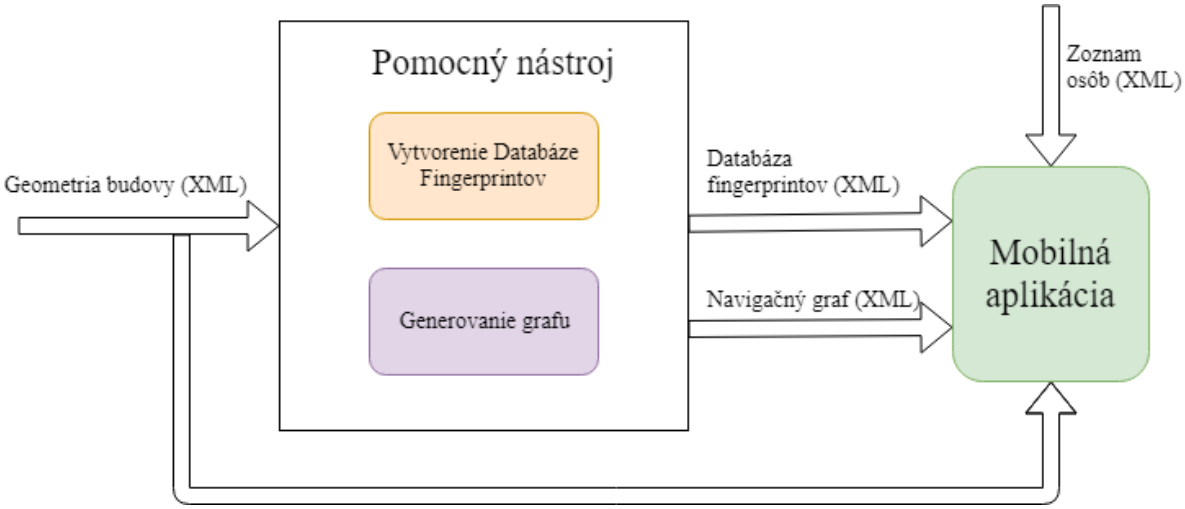
Pomocou získaných znalostí popísaných v predošlých kapitolách prichádza na rad návrh samotného postupu pre zjednodušenie aplikovania navigácie do budov. V tejto kapitole bude popísaný postup a jeho podrobné rozdelenie. Spomenuté budú taktiež metódy, ktoré bude postup využívať pri skenovaní a tvorbe grafu pre účely navigácie. Budú taktiež bližšie špecifikované podklady potrebné pre implementovanie lokalizačného a navigačného systému do budov na mobilnom zariadení.

4.1 Návrh postupu

Tvorba navigačného systému tvoria dve hlavné časti, lokalizovanie a vytvorenie navigácie. Pred samotným implementovaním je potreba sa zamyslieť nad získavaním dát na lokalizovanie a navigovanie a v čo najrozumnejšej forme tieto dáta budeme môcť postup aplikovať na rôzne iné budovy. Preto navrhujem postup, ktorý tvoria dve časti, pomocný nástroj na tvorbu lokalizačných databáz a generovanie grafovej štruktúry pre účely navigácie a mobilnej aplikácie, ktorá bude využívať výstupy z pomocného nástroja. Diagram postupu je zobrazený na obrázku 4.1.

Pomocný nástroj bude mať na vstupe geometrickú reprezentáciu definovanú napríklad v súbore typu XML (ukážka v sekcií 4.4). Pomocou tohto súboru sa vykreslí mapa, kde bude užívateľ vytvárať fingerprinty a generovať graf. Do mobilnej aplikácia sa vložia výstupy z pomocného nástroja pričom sa priložia súbory s definíciou mapy a dodatočné informácie o osobách, kde je definovaný vzťah medzi osobou a kanceláriou.

Pomocný nástroj na tvorbu podkladov pre úspornejšie implementovanie navigačného systémú do budov na mobilnom zariadení zastáva úlohu tvorcu databáze pre lokalizovanie a vytvorenie grafu pre účel vytvorenia navigačnej funkcionality v mobilnej aplikácií. Vstupnou hodnotu nástroja budú tvoriť definície geometrie poschodí budovy, ktorá umožní vykreslenie mapy do obrazovky nástroja. Obrazovka s mapu bude poskytovať možnosti interakcie s užívateľom v podobe vykonanie kliknutia, kedy sa vykoná vybraná funkcionalita. Funkcie poskytujúce v nástroji budú skenovanie a vytvorenie lokalizačnej databázy a generovanie grafu sprevádzané s dodatočným doplnením od užívateľa. Táto funkcionalita bude poskytovať výstup v podobe podkladov definovaných v XML súboroch (sekcia 4.4) pre ďalšie spracovanie v mobilnej aplikácií.



Obrázek 4.1: Diagram postupu tvoria dve časti a tou je pomocný nástroj, ktorý obsahuje funkcie získavania fingerprintov a generovanie grafu. Výstup z pomocného nástroja sa využije v mobilnej aplikácií.

Prípady užitia pomocného nástroja

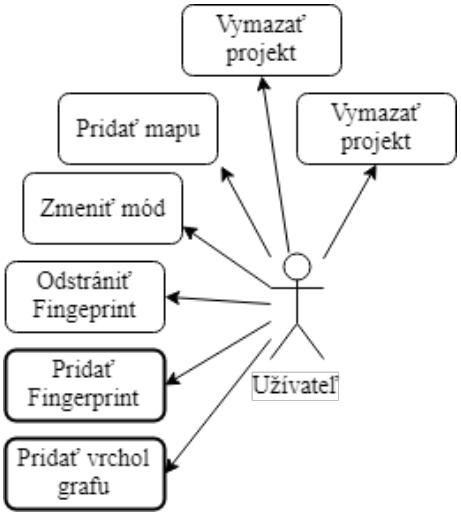
Vyjadrenie prípadov užitia je modelované pomocou diagramu prípadov užitia. Tento diagram je typ jazyka UML, ktorý je používaný pre návrh funkcionality aplikácie a definuje ako môže užívateľ s danou aplikáciou pracovať. Pred návrhnutím grafického rozhrania je dôležité, aby boli správne navrhnuté funkcie danej aplikácie a čo aplikácia má umožňovať. Modelovanie som uskutočnil pomocou online nástroja draw.io[1]. Na obrázku 4.2 je uvedený vytvorený diagram, z ktorého možno vidieť rôzne funkcie implementovaného nástroja, kde sa budú pravdepodobne najviac využívať tie prípady užitia, ktoré sú vyobrazené hrubšou čiarou.

4.2 Vytvorenie databázy fingerprintov

Základnou časťou každého navigačného systému je dobrá lokalizácia užívateľa v priestore. Vo vnútri budov však čelí najvyužívanejší systém pre určenie polohy – GPS (Global positioning system) výzve a tou sú steny, ktoré signály z vysielačov na obežnej dráhe Zeme pohlcujú a tak znemožňujú použiť tento systém k lokalizovaniu vo vnútri budovy. Preto bola venovaná pozornosť alternatívam, ktoré sú využiteľné v budovách v sekcií č. 2.

Výber som vymedzil na využitie zariadení vysielajúcich WIFI signál, kvôli využitiu už existujúcej infraštruktúre v budove, čiže nebude potreba vkladať ďalšie prostriedky do vybudovanie infraštruktúry. Nakoniec skončil výber na metóde WIFI fingeprinting pre jej presnosť voči ostatným metódam využívajúcich WIFI signál. Táto metóda však nesie časovú záťaž a to hlavne v offline fáze, kde je nutné získať informácie o okolitých access pointoch na určitej pozícií v mape nato, aby sa mohlo uskutočniť porovnanie a určenie polohy s fingerprintom z online fázy. Počas skenovania sa vložia do fingerprintov MAC adresy a korešpondujúce sily signálov z okolitých vysielačov. Takto vytvorený fingerprint sa

[1]Drawio – https://www.draw.io/



Obrázek 4.2: Diagram prípadov užitia pre nástroj umožňujúci vytvorenie databáze fingerprintov a generovanie grafu

zaznamená aj so súradnicami do databázy definovanej pomocou súboru vo formáte XML s definíciou popísanou v sekcií 4.4.

Zber dát do databázy fingerprintov bude prebiehať použitím pomocného nástroja. Priebeh bude taký, že užívateľ bude pomocou mapy zobrazenej v nástroji prechádzať budovu po častiach. Prechod budovy bude sprevádzať zastavovanie na miestach v určitej zrnitosti, ktorú si zvolí samostatne užívateľ a to miesto, na ktorom aktuálne užívateľ stojí vyberie v mape pomocou kliknutia. Po kliknutí sa spustí skenovanie, ktoré získa informácie o access pointoch a to hlavne silu signálu a MAC adresu zariadení v okolí státia užívateľa. Taktiež je možné využiť ku skenovaniu fingerprintov robota, nato však teraz nemáme prostriedky.

4.3 Tvorba grafu

Manuálna tvorba grafu pre možnosti navigácie nie je veľmi efektívna vo veľkej, komplexnej budove. Preto uvažujeme o spôsobe spomenutom v [16], ktorým túto časovo náročnú operáciu zjednoduší. Naskytá sa využiť spôsob geometrie a susednosti jednotlivých miestností, preto je potrebné získať rozmery a umiestnenia miestností v budove. Miestnosti budú definované pomocou vrcholov a ich súradníc. Takáto reprezentácia miesností je zobrazená v kóde XML č. 4.6. Preto aby sme mohli vytvoriť graf potrebujeme získať vrcholy grafu a zároveň hrany medzi nimi, ktoré budú ohodnotené. Týmto dvom podproblémom sa venujem v nasledujúcich riadkoch

Vytvorenie vrcholov

Vrcholy budem tvoriť najskôr od chodieb, ktoré sú základnou časťou budovy, po ktorých sa užívateľ najviac pohybuje. Základný údaj budú tvoriť stredy chodieb, ktoré vypočítame pomocou priemeru súradníc jednotlivých chodieb. Toto platí aj pre schodiská alebo výťahy. Stredové vrcholy chodby si nazvem ako hlavné vrcholy a stredové vrcholy schodísk a výťahov terminálové vrcholy pre prehladnejšie vysvetlenie ďalšieho postupu. Ďalej je nutné

rozpoznať vstupy do miestnosti, tie je možné získať manuálnym výberom užívateľa alebo automaticky pomocou vzťahov, ktoré obnášajú geometrickú reprezentáciu budovy. Ja som vybral automatické riešenie, kde využijem geometriu definovanú v súboroch, ktoré sa starajú o vyobrazenie mapy v nástroji. Z tohto vzťahu vieme získať pre všetky miestnosti úsečky, definujúce ich obvod. Úsečky budú prechádzané bod po bode pomocou Bresenhamovho algoritmu (kapitole 3), ktorý v pôvodnej implementácií vykresluje úsečky. Popri prechádzaní bodov miestnosti budem testovať, či sa daný bod nenachádza aj z jednej z chodieb. Ak sa nachádza poznačím si tento bod a budem pokračovať ďalej až dokým sa bod z obsahu chodby vytratí. Po tom, čo sa už bod nenachádza v chodbe si zoberiem posledný nachádzajúci sa bod v chodbe a poznačený prvý bod a vypočítam si priemer. Po výpočte priemeru získam stred medzi týmito bodmi a tento bod prehlásim ako vrchol grafu reprezentujúci vstup do miestnosti.

Pretože vyššie spomenuté postupy k vytvoreniu vrcholov sú nedostatočné vzhľadom k prehľadnosti a spojitosti grafu musia byť doplnené. Riešením môže byť zapojenie užívateľa do tvorby grafu, kedy si on sám zváži, kde by chcel mať ďalšie vrcholy nato, aby mohol vytvoriť dostatočne spojený graf.

Vytvorenie hrán medzi vrcholmi

U grafovej reprezentácie navigáčného systému je dôležité prepojiť jednotlivé miesta tak, aby boli vytvorené existujúce cesty medzi jednotlivými miestami na mape. Zníženie počtu nezmyselných hrán ako napríklad prechod cez stenu miestnosti je riadené pomocou grafu viditelnosti. Táto technika vytvorenia grafu je popísaná v kapitole s číslo 3. Graf viditelnosti požaduje definíciu objektov, ktoré budú chápané ako prekážky v ceste medzi jednotlivými vrcholmi. V prípade navigácie sú prekážky definované ako steny klasických miestností určené priamkami súradníc miestnosti, ktorej prináležia. Uvažujem, že jediné dovolené prechody sú tie, ktoré sa nachádzajú v chodbách. Využijem znovu Bresenhamov algoritmus (3), ktorým testujem jednotlivé body medzi vrcholmi grafu, vygenerované v predchádzajúcej časti tak, či prináležia niektorej z chodieb. Ak len jeden bod nie je situovaný v priestore chodby, hrana sa nevytvorí a nepridá do grafu.

Prepojenia medzi vstupmi do miestnosti a vrcholmi v chodbe by nemali byť prepojené s každou hranou. Ak by sa tak stalo, tak by v grafe vzniklo obrovské množstvo redundantných hrán a grafová štruktúra by zaberala veľké miesto v pamäti a tým pádom by bola spôsobená vyššia časová náročnosť pri vyhľadaní cesty. Množstvo hrán zredukujem pomocou implementácie nájdenia najkratšej hrany medzi vrcholmi vstupov a hlavnými vrcholmi chodby. Takto sa prejde celá množina hlavných vrcholov s tými vstupnými do miestnosti a vyberie sa tá, ktorá ma najmenšiu vzdialenosť, ktorá sa uloží do grafovej štruktúry. Prechody medzi chodbami sú taktiež spojené týmto spôsobom nájdenia najkratšej hrany. Týmto postupom získame jednoduchý graf, ktorý však nie je prehľadný a nie dostatočne spojený kvôli nedostatku hlavných vrcholov v niektorých chodbach a to hlavne v tých, s mnohouholníkovým tvarom.

Zvýšením počtu hlavných vrcholov je spomenuté vyššie, kde je navrhnuté manuálne pridávania užívateľom. Pridávaním vrcholov vzniká možnosť vytvorenia nových hrán. Pri pridaní je teda potreba prepočítať na novo spojenia medzi vstupmi a hlavnými vrcholmi chodby, pričom sa prepočítavajú len tie vrcholy, do ktorej bol pridaný vrchol užívateľom. Ďalšie hrany ktoré môžu vzniknúť sú tie medzi hlavnými vrcholmi chodieb.

Pri spájaní hlavných vrcholov chodieb však môžu vzniknúť redundantné cesty. Odstránenie redundancie hrán prebieha nasledovne. Vyberiem si z poľa vrcholov chodby dva hlavné

vrcholy *V1* a *V2*, ktorá bude reprezentovať hranu *E*. Ak existuje iná cesta ktorá vedie do *V2* z *V1* a táto cesta je tvorená z hrán, ktoré sú menšie ako hrana *E*, tak hranu *E* odstránime z grafu. Nižie je popísaná metóda pomocou pseudokódu č. 2 starajúca sa o redukovanie týmto spôsobom a je aj priložené zobrazenie vytvoreného postupu na obrázku č. 4.3.

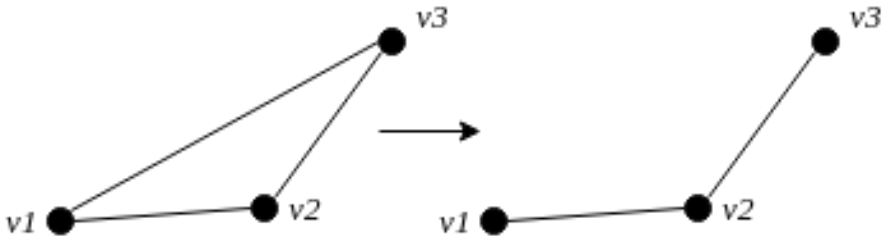
Algoritmus 2: Ukážka pseudokódu pre redukciu hrán v grafe

```
   Input  : mainVertices, edges
   Output: Reducedgraph
 1: allEdges = HashSet(Edge)
 2: seenVertices = HashSet(Edge)
 3: foreach Vertice outer in mainVertices do
 4:     foreach Vertice inner in mainVertices do
 5:         if (outer = inner) then
 6:             continue;
 7:         EdgeE = getEdgeByVertices(outer, inner)
 8:         if (E = NULL) then
 9:             continue;
10:         edges.remove(E)
11:         counter = 0
12:         if (!shortestPathExists(inner, outer) then
13:             edges.addBack(E);
14:             continue;
15:         foreach (k = 0; k < shortestPath.size()-1; k++) do
16:             Edge ex = Edge(shortestPath.get(k),shortestPath.get(k+1));
17:             if (e.getCost() >= ex.getCost()) then
18:                 counter + +;
19:         end foreach
20:         if (counter ≠ shortestPath.size()) then
21:             edges.addBack(E);
22:     end foreach
23: end foreach
```

4.4 Reprezentácia podkladov

Aby bolo možné vytvoriť pomocný nástroj a výslednú aplikáciu, ktorá umožní navigovanie a lokalizovanie je potreba navrhnúť správne dáta, ktoré sa budú získavať a spracovávať. Pri navigácií vo vnútorných priestoroch pracujeme s rôznymi dátami ako napríklad databáza fingerprintov pre lokalizovanie alebo geometrická reprezentácia budovy. Pre lepšiu manipuláciu a čitateľnosť som navrhol formát reprezentácie takýchto dát najmä v súboroch vo formáte XML. Ku každému navrhnutému súboru je k dispozícií priklád ukážky kódu, ale aj DTD[2], ktorý určuje povolené atributy a značky a je možné podľa tohto zápisu generovať XML súbory, ktoré budú spĺňať formátové požiadavky.

[2]Document Type Definition



Obrázek 4.3: Redukcia hrán v grafe. Na obrázku sa zruší hrana medzi vrcholmi *v1* a *v2*

Graf

Reprezentácia grafu môže byť reprezentovaná napríklad pomocou matice susednosti, ale v tejto práci je reprezentácia grafu definovaná značkovacím jazykom XML pre jeho jednoduchosť a dobrú čitateľnosť. Graf podľa definície, ktorá je vysvetlená v sekcií 3.2 musí obsahovať vrcholy a hrany. Informácie potrebnú o vrcholoch sú definované v značke `vertex`. Vrcholy ďalej nesú informácie o súradniciach, tie budú slúžiť pre vykreslovanie na mape. Ďalej je vo vrcholoch zahrnutá informácia o izbe, ktorú vrchol prepája s chodbou, inými slovami, ak nebude definovaná určitá miestnosť v atribute `roomID` (hodnota `null`), atribut bude určovať vrchol, ktorý nie je vrcholovým uzlom grafu a tak nereprezentuje dvere do miestostni. Jednoznačná identifikácia vrcholu je tvorená vo formáte `v_F_XXX_YYY`, kde `F` reprezentuje poschodie mapy, `XXX` reprezentuje x–ovú súradnicu a `YYY` y–ovú súradnicu. Hrany slúžia ako prepojenie medzi jednotlivými vrcholmi. V XML súbore sú grafy dané pomocou značky `edge`. Vyhľadávanie najkratšej cesty je podmienené tým, aby hrany medzi jednotlivými vrcholmi boli ohodnotené. Takúto informáciu nesia každá značka s názvom `edge`, ktorá je reprezentovaná atributom `cost`. V tejto značke reprezentujúcej hranu vrcholu sa uchovávajú informácie o zdrojovom a cieľovom vrchole grafu, tieto informácie sú zadané formou identifikátora vrcholu. V XML zdrojovom kóde č. 4.2 je možné vidieť deklarovanie grafu s uzlami `v_1_860_1934, v_1_887_1915, v_1_871_1934` a `v_1_908_1915`. Ďalej je uvedený kód č. 4.1 s DTD, ktorý definuje definíciou takto vytvoreného dokumentu.

```
<!DOCTYPE Graph [
  <!ELEMENT Graph (vertex*,edge*)>
  <!ELEMENT vertex EMPTY>
  <!ELEMENT edge EMPTY>

  <!ATTLIST vertex x CDATA #REQUIRED>
  <!ATTLIST vertex y CDATA #REQUIRED>
  <!ATTLIST vertex roomID CDATA #REQUIRED>
  <!ATTLIST vertex id CDATA #REQUIRED>
  <!ATTLIST edge src CDATA #REQUIRED>
  <!ATTLIST edge dst CDATA #REQUIRED>
  <!ATTLIST edge cost CDATA #REQUIRED>
]>
```

Kód 4.1: Definícia typu dokumentu XML popisujúca zoznam osôb.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<Graph>
  <vertex y="1934" x="860" roomID="Q116" id="v_1_860_1934"/>
  <vertex y="1915" x="887" roomID="Q114" id="v_1_887_1915"/>
  <vertex y="1934" x="871" roomID="null" id="v_1_871_1934"/>
  <vertex y="1915" x="908" roomID="Q113" id="v_1_908_1915"/>
  <edge src="v_1_860_1934" dst="v_1_887_1915" cost="720.0"/>
  <edge src="v_1_887_1915" dst="v_1_871_1934" cost="60.0"/>
  <edge src="v_1_887_1915" dst="v_1_908_1915" cost="540.0"/>
  <edge src="v_1_908_1915" dst="v_1_860_1934" cost="320.0"/>
</Graph>
```

Kód 4.2: Ukážka definovania grafu pomocou XML. Prvok vertex reprezentuje uzol grafu obsahujúci atributy koordinátov (x,y) identifikátor miestností (roomID) a samotný identifikátor vrcholu grafu (id) pre jednoduchšie referencovanie pri hranách grafu. Značka edge reprezentuje hranu, ktorá nesie informáciu o hodnote hrany atributom cost a vrcholy tvoriace hranu v atribute src a dst.

Databáza Fingerprintov

Pre vytvorenie databáze fingerprintov taktiež poslúži značkovací jazyk XML pre vyššie uvedené výhody. Príklad definície je uvedený v kóde č.4.4, ktorý je generovaný na základe DTD kódu č. 4.3. Takto reprezentované dáta budú prevedené do triednej reprezentácie, kde každá časť xml má svoju triedu, v ktorej budú implementované funkcie potrebné pre lepšiu prácu s datami. Fingerprint musí obsahovať vektor usporiadaných dvojíc MAC adries a k ním prináležitých síl signálov. Vektor je reprezentovaný pomocou značiek s označením `AP` s atribútmi *mac* a *strength*. Tieto vektory sú zaobalené v značke `fingerPrint`, ktorá udáva súradnice *x,y* a identifikátor fingerprintu *id*. Ďalej sú fingerprinty priradené jednotlivému poschodiu, kvôli identifikácií čísla poschodia, tie sú reprezentované v značke `Map` obsahujúca číslo poschodia, pričom v súbore ich môže byť viacero.

```
<!DOCTYPE FingerprintMap [
  <!ELEMENT FingerprintMapFingerprintMap (Map*)>
  <!ELEMENT Map (fingerPrint*)>
  <!ELEMENT fingerPrint (AP*)>
  <!ELEMENT AP EMPTY>

  <!ATTLIST Map floor CDATA #REQUIRED>
  <!ATTLIST fingerPrint x CDATA #REQUIRED>
  <!ATTLIST fingerPrint y CDATA #REQUIRED>
  <!ATTLIST fingerPrint id CDATA #REQUIRED>
  <!ATTLIST AP strength CDATA #REQUIRED>
  <!ATTLIST AP mac CDATA #REQUIRED>
]>
```

Kód 4.3: Definícia typu dokumentu XML popisujúca súbor definujúci databázu fingerprintov.

```xml
<?xml version="1.0" encoding="utf-8"?>
<FingerprintMap>
  <Map floor="1">
    <fingerPrint y="1183" x="1183" id="0">
      <AP strength="-64.8" mac="64:7c:34:a2:6e:2f"/>
      <AP strength="-74.0" mac="ac:22:05:a6:2d:c3"/>
      <AP strength="-82.4" mac="90:5c:44:f9:fd:4d"/>
    </fingerPrint>
  </Map>
</FingerprintMap>
```

Kód 4.4: Príkladná dafinícia súboru s fingerprintami. Na obrázku je možno vidieť mapu s poschodím jedna a v tomto poschodí jeden fingerprint so súradnicami x = 1183 a y = 1183 a identifikátorom nula. Fingerprint obsahuje tri MAC adresy s príslušnými tromi silami signálov.

Reprezentácia budovy

Súbor obsahujúci definíciu geometrie budovy obsahuje tri značky a to: `map, room` a `coord`. Značka map nesie atribut s označením `floor`, ktorý určuje číslo poschodia tejto mapy. Ďalej značka `room` obsahuje atributy `geometry, name, idRoom` a `type`. Tieto `atributy` určujú geometriu miestnosti a to či je miestnosť polygoniálneho tvaru alebo štvoruholníkového tvaru, `name` nesie informácie o mene miestnosti a `idRoom` jedinečnú identifikáciu miestnosti v celej budove. Atribut `type` môže nadobudnúť tri hodnoty: room, hall a terminal čo znamená že miestnosť je buď normálna izba alebo kancelária v prípade typu *room*, chodba určená pomocou *hall* a prechod, dvere. Schody alebo výťah určuje typ *terminal*. Poslednou značkou je `coord`, ktorá určuje x–ové a y–ové súradnice rohov miestnosti. Pre lepši predstavu je priložená ukážka takto vytvoreného súboru definujúceho mapu s jednou miestnosťou na zdrojovom kóde 4.6.

```
<!DOCTYPE map [
  <!ELEMENT map (room*)>
  <!ELEMENT room (coord+)>
  <!ELEMENT coord EMPTY>

  <!ATTLIST map floor CDATA #REQUIRED>
  <!ATTLIST room geometry (RECT|POLY) CDATA "RECT">
  <!ATTLIST room name CDATA #REQUIRED>
  <!ATTLIST room idRoom CDATA #REQUIRED>
  <!ATTLIST room type (hall|room|terminal) "room">
  <!ATTLIST coord x CDATA #REQUIRED>
  <!ATTLIST coord y CDATA #REQUIRED>
]>
```

Kód 4.5: Definícia typu dokumentu XML popisujúca súbor s geometrickou reprezentáciou mapy.

```xml
<?xml version="1.0" encoding="utf-8"?>
<map floor="1">
  <room geometry="RECT" name="Kitchen" idRoom="Q101" type="room">
    <coord x="438" y="910"/>
    <coord x="460" y="910"/>
    <coord x="460" y="940"/>
    <coord x="438" y="940"/>
  </room>
</map>
```

Kód 4.6: Príklad definície mapy s poschodím číslom 1 a jednej miestnosti štvoruholníkového tvaru s názvom Kitchen, jednoznačným identifikačným údajom Q101 a typom room. Pričom geometria miestnosti je daná súradnicami v značkách coord

Zoznam osôb

Nato aby bolo umožnené užívateľovi vyhľadať osoby, ktoré sídlia v budove v určitej kancelárií je nevyhnutné vytvoriť zoznam takýchto osôb. XML súbor definujúci zoznam osôb obsahuje koreňový uzol, ktorý tvori značka `people`. Po koreňovej značke sú následne vetvené ďalšie značky s pomenovaním `person`, ktoré nesú informácie o mene, kancelárií alebo mena oddelenia v atributoch `departments, name` a `room`. Zdrojový text č. 4.8 je priložený pre príkladnú definíciu takto vytvoreného súboru aj so všeobecnou definíciou typu takéhoto dokumentu v kóde č.4.7. Takéto dáta budú poskytnuté užívateľom, ktorý chce vytvoriť navigačný systém.

```xml
  <!DOCTYPE people [
    <!ELEMENT people (person*)>
    <!ELEMENT person EMPTY>

    <!ATTLIST person name CDATA #REQUIRED>
    <!ATTLIST person department CDATA #REQUIRED>
    <!ATTLIST person room CDATA #REQUIRED>
  ]>
```

Kód 4.7: Definícia typu dokumentu XML popisujúca zoznam osôb.

```xml
<?xml version="1.0" encoding="utf-8"?>
<people>
  <person department="UITS" name="Novotny Peter, Ing." room="A223"/>
  <person department="VCIT" name="Nezabudkova Radka, Mgr." room="Q199"/>
  <person department="UPGM" name="Bambusovy Peter" room="P1505"/>
</people>
```

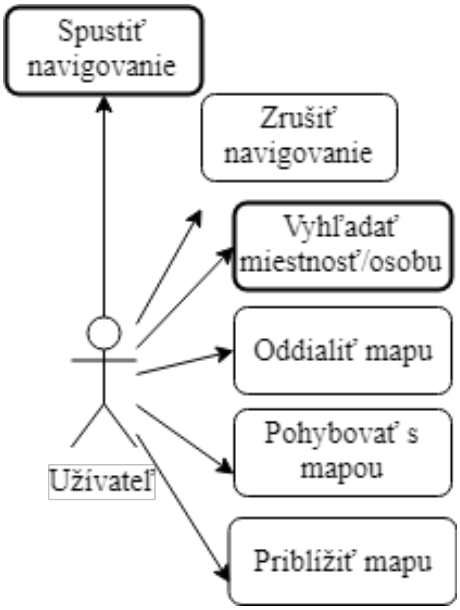
Kód 4.8: Príklad definície mapy s poschodím číslom 1 a jednej miestnosti štvoruholníkového tvaru s názvom Kitchen, jednoznačným identifikačným údajom Q101 a typom room. Pričom geometria miestnosti je daná súradnicami v značkách coord

4.5 Mobilná aplikácia na navigovanie

Mobilná aplikácia bude využívať jednotlivé XML súbory, ktoré užívateľ manuálne vloží do projektu s vývojom aplikácie. Súbory definujúce zoznam osôb budú využité pre prehľadávanie spolu s geometrickou reprezentáciou, ktorá bude taktiež použitá na vykreslovanie mapy na obrazovku zariadenia. Mapa bude interaktívna na základe dotykov užívateľa, kedy užívateľ bude môcť priblížiť, oddialiť, pohybovať alebo vyberať miesto určenia. Databáza fingerprintov bude spracovaná a využitá pre lokalizovanie pomocou metódy fingerprinting a nakoniec do aplikácie bude vložený XML súbor určujúce graf, ktorý bude reprezentovať navigačné cesty po celej mape.

Prípady užitia

Najfrekventovanejšie používanou funkciou mobilnej aplikácie bude nepochybne lokalizovanie a vyhľadávanie osoby alebo miestnosti. Hlavne pre tieto úkony bude užívateľ aplikáciu používať. Tieto a ďalšie funkcie, ktoré by mala aplikácia obsahovať sú zobrazené v diagrame na obrázkoch číslo 4.4.



Obrázek 4.4: Diagram prípadov užitia pre mobilnú aplikáciu.

4.6 Návrh užívateľského rozhrania mobilnej aplikácie

Návrh užívateľského rozhrania je dôležitou časťou každej aplikácia či sa jedná o mobilné, webové alebo desktopové aplikácie. Na začiatku by mal byť nákres našej predstavy o aplikácií na papier – *wireframe*. Wireframe je možné jednoduhco upravovať alebo začať znovu iným papierom. Po premyslení všetkých častí by malo nasledovať prekreslenie Wireframu do digitálnej podoby. Ja som zvolil nástroj *proto.io*[3]. Proto.io je online webový nástroj pre pro-

[3]Proto.io – https://proto.io/

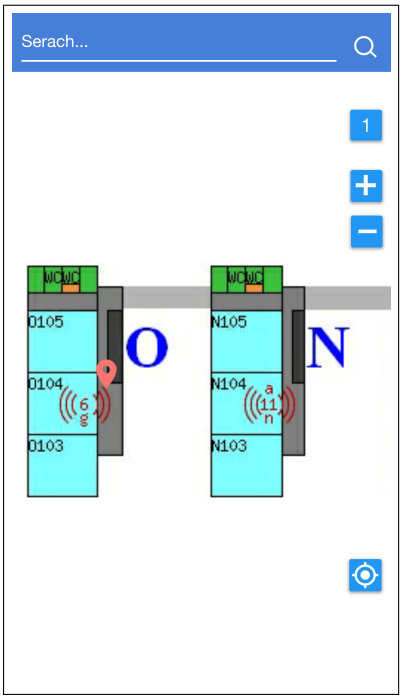
totypovanie grafického rozhrania. Umožňuje vyhľadať jednotlivé grafické prvky a pomocou funcionality drag and drop tieto prvky usporiadať podľa prvotného wireframe návrhu.

Hlavná obrazovka

Po spustení sa vždy objaví hlavná obrazovka užívateľovi. Preto by mala byť jednoduchá a intuitívna. Základným prvokom celej hlavnej obrazovky je zobrazenie mapy.Toto zobrazenie tvorí vizuálne prepojenie s lokalizáciou a samotnou navigáciou, kde sa budú zobrazovať vyhľadané cesty alebo poloha užívateľa na mape.

Pre jednoduché vyhľadávanie je na hlavnej obrazovke umiestnené tlačidlo s motívom lupy. Znak lupy som zvolil pre jej rozšírenie, kedy sa vo väčšine aplikáciách, kde je možné vyhľadávanie objavuje ako znak pre vyhľadanie určitej informácie v databáze. Po kliknutí sa zobrazí textové pole, kde užívateľ zadá výraz, ktorý bude následne vyhľadaný a budú zobrazené výsledky vyhľadávania.

Typickými tlačidlami v aplikáciách určených pre navigovanie vo vnútorných priestoroch sú napríklad zmena poschodia, priblíženie a oddialenie zobrazenia mapy alebo lokalizácia užívateľa. Všetky zmienené príklady sú zahrnuté v návrhu na obrázku 4.5.



Obrázek 4.5: Wireframe hlavnej obrazovky aplikácie

Kapitola 5

Nástroj na vytvorenie podkladov

Kapitola zhrňuje postup implementácie nástroja pre zbieranie vzorkov pre potreby lokalizácie a súčastne aj generovanie grafu z geometrickej reprezentácie budovy. V úvode bude reč o grafickom rozhraní desktopovej aplikácie a o využitých grafických prvkoch, ktoré sú v nástroji použité. Ďalej budú spomenuté očakávané vstupy a výstupy pre túto aplikáciu. Pre zbieranie síl signálov a príslušných MAC adries je implementovaná a sú vysvetlené implementačné detaily metódy, ktorá sa o túto funkciu stará. V časti 5.3 bude popísený postup implementácie generovania grafu z geometrickej reprezentácie mapy budovy. V kapitole je zahrnuté aj testovanie vplyvu času na fingerprinty v databázi.

Vstupy

Vstupom pre nástroj je definícia mapy. Táto mapa je definovaná pomocou XML súbora, ktorý reprezentuje mapu poschodia. Príklad takejto reprezentácie je uvedený v kapitole 4.

Pre demonštráciu aplikácie bola stiahnutá definícia budovy Fakulty informačných technológií v Brne. Mapa fakulty je uvedená na webových stránkach[1], kde je generovaná z informačného systému. Pre spracovanie webových stránok som využil knižnicu `BeautifulSoup` v jazyku `python`. Knižnica umožňuje rozobrať zdrojové kódy webových stránok podľa značiek a atributov jazyka `HTML`. Získaný kód je spracovaný tak, že sú vybraté len tie informácie (názov budovy, súradnice), ktoré sú relevantné a následne uložené do `XML` súboru v už spomenutom formáte.

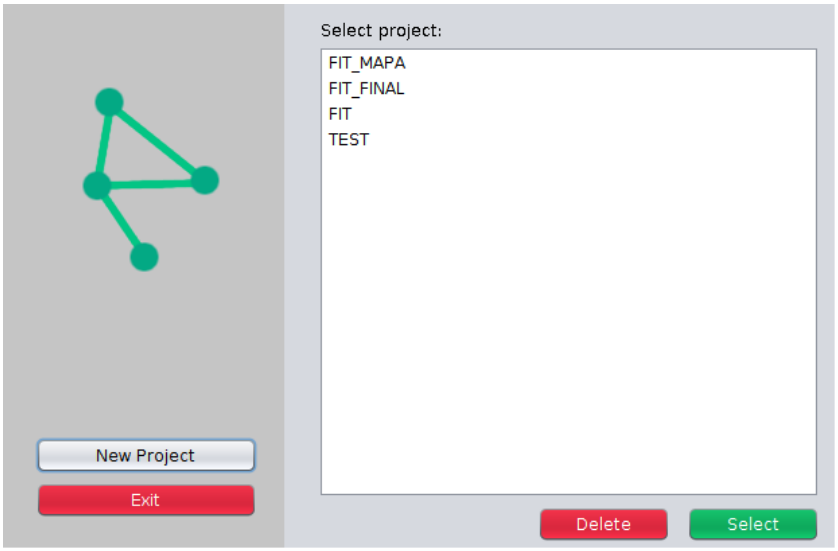
5.1 Užívateľské rozhranie

Grafické rozhranie som rozdelil na dve časti a to na úvodnú obrazovku, kde bude môcť užívateľ vytvoriť projekt s jeho mapou a na hlavnú obrazovku, kde bude užívateľ môcť generovať graf pre jeho definovanú mapu alebo vytvoriť databázu fungeprintov pre lokalizovanie. Pri vytváraní grafického rozhrania som vychádzal z diagramu prípadov užitia, ktorý je spomenutý v kapitole 4.

[1] Stránka s mapou fakulty http://www.fit.vutbr.cz/FIT/map/fit2.php.cs

Úvodná obrazovka

Úvodná obrazovka slúži na uvítanie užívateľa, ktorý si chce vytvoriť nový projekt s generovaním grafu alebo dát pre databázu fingerprintov alebo spustiť už v minulosti vytvorený projekt. Preto aby užívateľ mohol vybrať už existujúci projekt, je v grafickom rozhraní uvedený prvok implementujúci textový zoznam mien projektov pomocou `JListView`. Po výbere zo zoznamu si stlačením tlačidla `Choose` vyberie a spustí projekt alebo výberom tlačidla `Delete` vymaže celý projekt. Ďalej sú tu uvedené tlačidlá ako napríklad `New Project` a `Exit`, ktoré sú realizované pomocou JButton komponenty knižnice `Swing` a ktoré majú jednoznačne identifikovateľnú funkcionalitu. Na obrázku č.5.1 je možné vidieť výstup tejto úvodnej obrazovky.



Obrázek 5.1: Úvodná obrazovka pomocného nástroja

.

Hlavná obrazovka

Po vytvorení alebo vybere už existujúceho projektu sa zobrazí hlavná obrazovka. Táto hlavná obrazovka obsahuje hlavnú časť a tou je zobrazená mapa, ktorá je vykreslovaná pomocou polygónov. Užívateľovi je umožnené klikať na mapu a na základe módu sa po kliknutí vykoná príslušná funkcionalita. Mód určuje či sa majú po vybratí skenovať access pointy v okolí, ktoré sa následne uložia do databázy fingerprintov alebo sa má vložiť nový uzol do navigačného grafu. Realizácia možnosti zmeny je implementovaná jednoduchým `JRadioButton`, kedy zaškrtnutá možnosť určuje túto funkciu. Hlavná obrazovka ponúka možnosť zmeny zobrazeného poschodia mapy budovy pomocou `JTree`, ktorá hierarchicky zobrazuje poschodia, ktoré užívateľ vložil do tohto projektu. Samotné vloženie sa realizuje pomocou tlačidla so znakom plus, kedy sa užívateľovy zobrazí formulár, kde si z `JFileChooser` nástroja vyberie `XML` súbor definujúci požadované poschodie. V grafickom rozhraní sú obsiahnuté aj tlačidlá ako `Exit` alebo `Save` s motívom diskety.

5.2 Zbieranie fingerprintov

Po zaškrtnutí módu s názvom fingerprint je spustená funkcionalita zberu dát pre vytvorenie databáze fingerprintov. Táto funkcia po kliknutí na príslušné miesto na mape spustí zber informácií o sile a MAC adrese dosiahnuteľných access pointoch.

Algoritmus pre zber požadovaných dát využíva nástroj `iwlist`[2], čo je systémový nástroj pre Unix, ktorým je môžné získavať detailnejšie informácie ohľadom bezdrôtového rozhrania počítača. Nástroj je využitý v skripte `mapa.py`, ktorý je napísaný v jazyku `Python`. Skript spracováva výstup z nástroja `iwlist` tak, že vyhľadá len tie časti, ktoré sú zaujímavé pre uloženie do databáze fingerprintov. V prvom prípade vyhľadá všetky MAC adresy a im príslušné hodnoty signálov, tento proces sa opakuje 4 krát s odstupom 1 sekundy. Počas opakovania si skript ukladá do asociatívneho poľa MAC adresy a príslušné sily signálov k týmto adresám. V poli sa indexuje pomocou kľúča tvoreného MAC adresami. Po ukončení skenovania sa zoberie toto asociatívne pole a pre každý kľúč vypočíta priemernú hodnotu. Túto hodnotu ďalej predá programu, ktorý tento výstup požaduje.

Výstup skriptu spracováva implementovaný nástroj a uloží si ho do triednej reprezentácie pre každý fingerprint. Nevýhodou je, že `iwlist` neposkytuje presné informácie, ak nie je spustený pod hlavičkou užívateľa správcu zariadenia. Takže po každom spustení skriptu sa užívateľovi objaví formulár pre zadanie hesla k správcovskému prístupu k počítaču.

Po úspešnom preskenovaní dostupných access pointov je na obrázok mapy pridaný bod, ktorý reprezentuje vytvorený fingerprint a je tak vytvorený nový objekt triedy `FingerPrint`. Vytvorená trieda pridáva funkcionalitu klikania na vykreslene objekty reprezentované pomocou komponenty `JLabel` a nastavenia tejto funkcie s metódou `setOnClickListener`, kedy je vytvorený vlastný `ClickListener`, ktorý umožňuje pomocou pravého tlačidla zobraziť menu, kde si užívateľ môže vybrať jednu z dvoch možností. Tieto možnosti implementujú preskenovanie na novo už vytvoreného fingerprintu, kedy sa vybranému fingerprintu priradia nové hodnoty MAC adries a príslušných síl signálov alebo druhá možnosť výberu, vymazanie fingerprintu.

Uloženie databáze prebieha pomocou parsovania dát z poľa fingerprintov, uložených v aplikácií v triede `FingerPrintMap`, pomocou `SAXBuilder`. Táto knižnica poskytuje možnosť vytvoriť súbor vo formáte `XML` a špecifikovať potrebné dáta do tohto formátu. Príkladný vystup z uloženia databáze fingerprintov je uvedený v sekcií číslo 4.4.

5.3 Generovanie grafu

Vytvorenie grafu je dôležitou časťou tejto práce, pretože bez grafu by nebolo možné vykresliť a nájsť najkratšiu cestu medzi cieľovým bodom a koncovým bodom. V nasledujúcich riadkoch je popísaný implementácia postupu navrhnutého v kapitole 4 na získanie podkladov pre možnosť navigácie vo vnútorných priestoroch budovy.

Tvorba vrcholov

Budova je rozdelená do miestností a tie sú reprezentované pomocou triedy `Room`, ktorá obsahuje metódy ako `doesPolygonIntersectPolygon, reduceEdges` alebo `centroid`, ktoré umožňujú otestovanie či polygon pretína iný polygón, zredukovanie hrán v miestnosti a repsektíve výpočet centrálneho vrcholu pre danú miestnosť. Pri načítaní poschodia z XML

[2] iwlist – https://linux.die.net/man/8/iwlist

prebieha vytvorenie takýchto objektov pre všetky miestnosti v mape, kde je ďalej špecifikovaný typ týchto miestností (*room, hall, terminal*) a zároveň pri vzniku sa vypočítajú aj centrálne vrcholy pre tieto miestnosti a vzniknutý vrchol si uložia do grafu, ktorý prináleží danej miestnosti.

Proces tvorby grafu pre navigácie v pomocnom nástroji na tvorbu materialov začína prekliknutím módu na tvorbu grafu pomocou JRadioButton v pravej časti obrazovky nástroja. Ihneď po zmene módu sa spustí výpočet vrcholov grafu pre jednotlivé miestnosti v práve vybranej mape z budovy. Začiatočným krokom je dovypočítanie zvyšných vrcholov a to tých, ktoré reprezentujú vstupy do miestností. Pomocou prechodu cez všetky miestnosti typu *room*, kedy sa testuje, či hrany z týchto miestností nemajú spoločné body z hrán z miestností typu *hall*. Testovanie vykonávam pomocou Bresenhamovho algoritmu, ktorým algoritmus prejde všetky body medzi vrcholmi a tieto body testuje voči polygónu chodby pomocou metódy `contains`, ktorá zisťuje, či sa bod nachádza v polygóne. Ak sa bod v polygóne nachádza poznačím si do pomocnej premennej jeho koordináty a pokračujem takto ďalej až dokým sa bod nebude nachádzať vo vybranom polygóne, tak sa tento posledný bod zoberie a spriemeruje sa s pomocou premennou udávajúcou počiatočný bod. Týmto spôsobom získam stred hrany, ktorý si uložím do poľa vrcholov s dodatočnou informáciou, ktorá uvádza identifikátor miestnosti, do ktorej vstup patrí.

Ako bolo spomenuté pri návrhu postupu, bude vždy dochádzať k nedostatku vrcholov, pomocou ktorých bude možné vytvoriť graf. Preto som navrhol, aby bol pripojená činnosť užívateľa k pridávaniu dodatočných vrcholov. Pridávanie je zabezpečené tým, že som implementoval do zobrazenia mapy `OnClickListener`, ktorým užívateľ po kliknutí vytvorí nový vrchol, ktorý sa priradí miestnosti do ktorej patrí podľa súradníc, pričom je pridávanie obmedzené len na chodby a terminálne miestnosti.

Tvorba hrán grafu

Po nadobudnutí všetkých vrcholov môže byť spustená implementácia získavania spojení medzi týmito vrcholmi. Spájanie vrcholov prebieha tak, že si v prvom rade zoberiem všetky chodby a terminály a spojím ich vstupy od iných miestností s najbližším hlavným vrcholom, ktorý je v týchto miestnostiach k dispozícií. Po spojení dverí a hlavných vrcholov prichádza na rad užívateľ, ktorý pomocou `OnClickListener` pridáva hlavné vrcholy do chodieb alebo schodísk. Po kliknutí algoritmus zisťuje aká chodba prináleží kliknutému miestu pomocou metódy `contains` volanej nad polygónom chodby. Po zistení do akej chodby sa vytvoril nový vrchol prebehne prepočet spojení a to hlavných vrcholov s dverami vybranej chodby. Ďalej sú pospájané samotné hlavné vrcholy ale narozdiel od Všetkých spájaniach kontrolujem pomocou Bresenhamovho algoritmu to, či sa spojenie medzi dvoma vrcholmi nachádza len v chodbách alebo schodiskách. Ak sa tak nestane a je toto spojenie chybné takáto hrana sa nevytvorí.

Posledným krokom pri tvorbe grafu je vytvorenie prepojenia medzi jednotlivými poschodiami. Pri riešení som vychádzal z predpokladu, že väčšina schodísk, ktoré sú vykreslované na mape sú približne rovnako umiestnené o poschodie vyššie alebo nižšie. Testovanie prebieha tak, že si zoberiem všetky hlavné vrcholy terminálnych miestností a postupne testujem, či sa koordináty hlavného vrcholu nenachádzajú z niektorej z terminálnych polygónov o poschodie vyššie pomocou metódy `doesPolygonIntersectPolygon`. Ak sa tak stane vytvorí sa nová hrana, ktorej velkosť je definovaná konštantou 50. Táto hodnota je daná konštatne pretože z geometrickej reprezentácie vnútorného priestora neexistuje vzťah,

ktorým by som vypočítal vzdialenosť medzi vrcholmi z iných poschodí a nie je tak možné vypočítať vzdialenosť na základe Euklidovskej vzdialenosti dvoch bodov.

Cesty, o ktorých užívateľ vie, že existujú a nie sú zobrazené, vzniká potreba vytvoriť tieto hrany manuálne. Vytvorenie hrany je zabezpečené pomocou kliknutia na jednu a druhú hranu, kedy sa vytvorí táto užívateľom požadovaná hrana. Funkcionalita môže poslúžiť aj pre prepojenie niektorých častí grafu, ktoré sú separovné od ostatných napríklad oddelené budovy.

Zobrazenie grafu

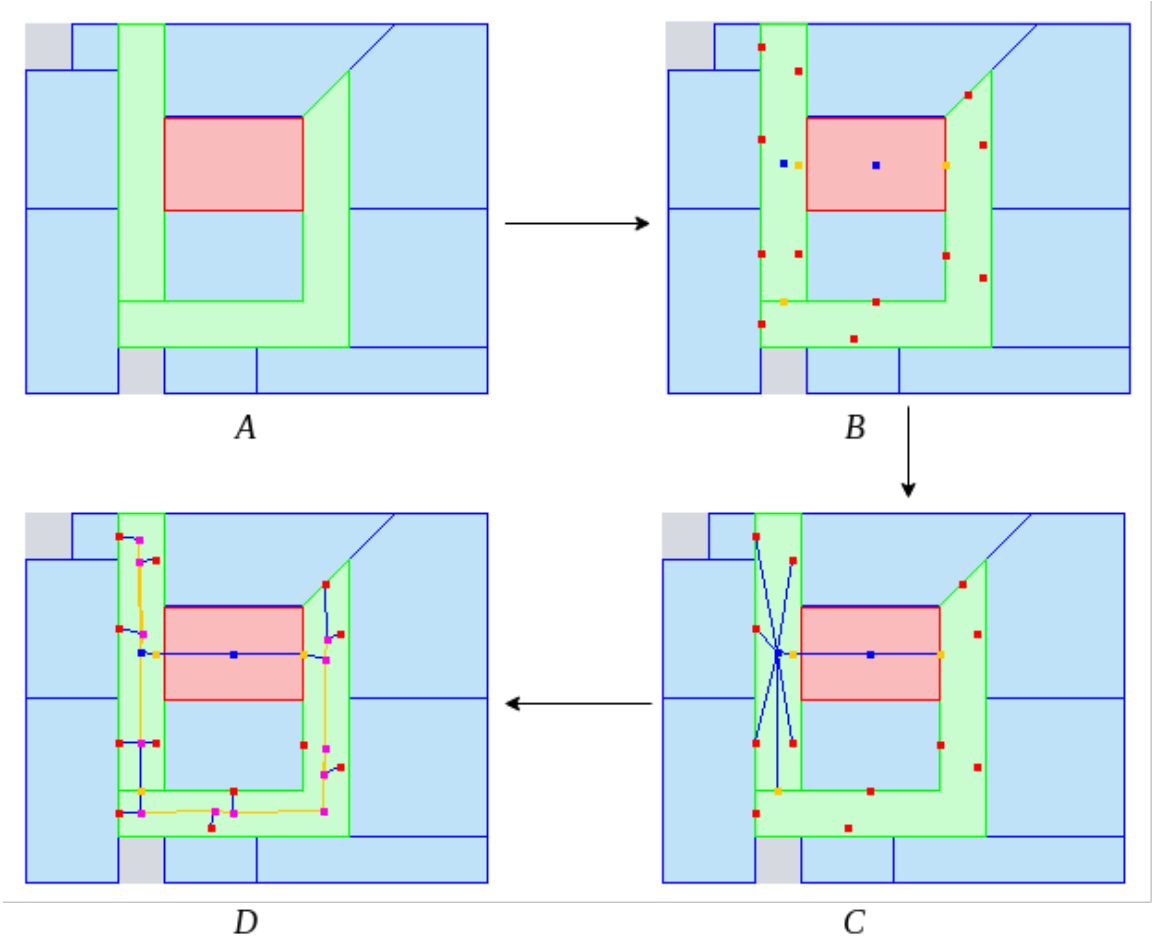
Graf je vykreslený pomocou funkcií `drawLine`, ktorá vykresluje hrany a `drawPoint`, ktorá vykresluje vrcholy grafu. Farby vrcholov identifikujú tri typy vrcholov, ktoré sú uvedené vyššie a to k červenej farbe prináležia vstupy do klasických miestností, oranžové značia prechody medzi chodbami, modré zase hlavné vygenerované vrcholy a nakoniec ružovou farbou sú vyznačené pridané vrcholy užívateľom.

Postupný vývoj, zobrazenie a vytvorenie grafu môžete vidieť na obrázku s číslom 5.2 pre lepšiu predstavu. V tomto obrázku sú uvedené štyri etapy, vďaka ktorým je získaná grafová štruktúra. Prvú etapu s označením A tvorí samostatná geometrická reprezentácia budovy. Etapa B zobrazuje vznik vrcholov vďaka susedným vzťahom miestností, kde body s červenou farbou tvoria vstupné miesta do miestností, modrou zase hlavné vrcholy chodieb a schodísk, žltou farbou je reprezentované spojenie medzi chodbami alebo schodiskami. Ďalšou etapou (C) je vyobrazené prvotné automatické generovanie hrán medzi vrcholmi a nakoniec je graf doplnený celý pomocou užívateľa, ktorý pridal ďalšie potrebné vrcholy pre sprehladnenie a dospájanie celého grafu, kde pridané vrcholy majú ružovú farbu.

5.4 Meranie časovej závislosti fingerprintov

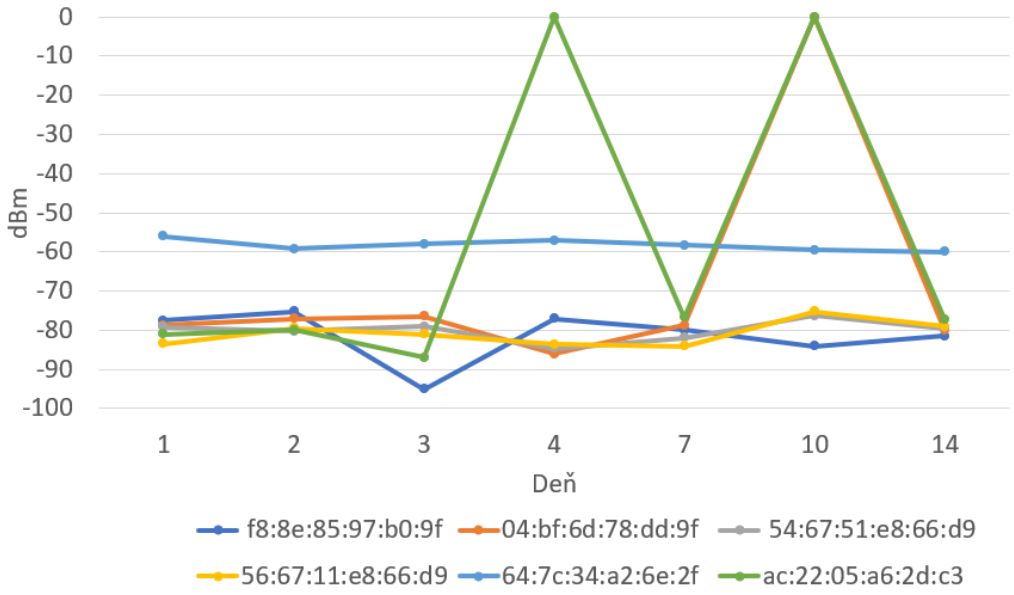
Pomocný nástroj som využil pri skenovaní Fakulty informačných technológií a vytvorení databáze fingerprintov. Skenovanie prebiehalo trvalo v čistý čas s aj s presunmi cca 2 hodiny kedy som za tento čas vytvoril 320 fingerprintov, čo vychádza na 22.5 sekundy na jeden fingerprint. Skenovaním som získal komplexnú databázu, ktorá je ďalej využiteľná v mobilnej aplikácií popísanej v kapitole 6. Skenovanie ukázalo, že v niektorých priestoroch budovy sa nachádza nedostatok, respektíve veľmi málo access pointov. Počet access pointov ovplyvňuje presnosť a čím je ich viac, tak tým je presnosť lepšia. Avšak počte 2 – 3 access pointy je nedostatočný. Takýto počet by však nebol vhodný v žiadnej metóde, ktorá využíva WIFI signál k lokalizovaniu. Takýchto miest je len málo a väčšinou prevládajú fingerprinty s počtom 6 – 9, ale nájdu sa aj miesta s 15 a viac access pointami.

Jednou z nevýhod metódy WIFI fingerprinting je aj to, že počas dynamického života v zmapovanej budove môžu nastať určité zmeny, preto by som rád získal skutočné čísla o tomto probléme. Počas testovania budem stále na rovnakom mieste skenovať dáta do fingerprintov počas 2 týždňov, pričom budem postupovať v prvý deň, druhý deň, štvrtý deň, siedmy deň, desiaty deň a nakoniec štrnásty deň. Vďaka týmto testom budem môcť určiť, aký veľký vplyv má čas respektíve prípadná zmena priestoru na fingerprinty a tak ovplyvnenie presnosti vybranej metódy pre lokalizovanie. intami. Počas vykonávania testovania som zistil, že určité výkyvy, ktoré môžu ovplyvniť presnosť metódy sa počas skenovania objavujú. Na obrázku 5.3 je možno vidieť šesť access pointov s príslušnými silami signálov v určitých dňoch. Z grafu je vidieť, že sa počas skenovania objavili skoky, ktoré sa dostali na



Obrázek 5.2: Postupné vytvorenie grafu pomocou geometrie miestností. Na obrázku môžete vidieť 4 fázy budovania grafu *(A, B, C, D)*.

hodnotu 0, čo znamená, že neboli detekované pomocou nástroja. Toto nedetekovanie access pointov môže jednoznačne negatívne ovplyvniť presnosť lokalizovacej metódy. Ďalšie oskenované access pointy vykazujú pomerne menšie skoky avšak určite aj tie dokážu znepresniť lokalizovanie avšak nie tak rapídne ako nedetekovanie.



Obrázek 5.3: Graf reprezentujúci výkyvy sily signálov (dBm) pre jednotlivé access pointy v čase (dňoch).

Kapitola 6

Mobilná navigácia do budov

Nasledujúcej kapitole bude podrobnejšie rozobratá implementácia mobilnéj aplikácie umožňujúcej navigovanie a lokalizovanie. V prvej podkapitole bude popísaná implementácia grafického rozhrania a použité grafické komponenty na jeho vytvorenie. V ďalšej bude nasledovať popis lokalizácie a na koniec implementácia navigácie a jej zobrazenie na mape.

6.1 Implementácia užívateľského rozhrania

Užívateľské rozhranie mobilnej aplikácie som implementoval na základe wireframu uvedeného v kapitole 4 s drobnými úpravami, hlavne čo sa týka zobrazenia vyhľadávanej miestnosti. Rozhranie som implementoval pomocou grafických prvkov, ktoré sú deklarované v XML súboroch. Takýmto spôsobom Android oddeľuje front-end a back-end. V každom takomto XML súbore je možné deklarovať vlastnosti grafických prvkov, akými sú umiestnenie na obrazovke, farba alebo tvar. Na implementáciu týchto súborov sú využité predovšetkým *ConstraintLayout*[1], ktorý umožňuje flexibilne získať polohu a zväčšovať komponenty. Okrem tohto layoutu boli využité *LineraLayout* alebo *FrameLayout*. Keďže sa jedná o navigačnú aplikáciu, nie sú nutné žiadne náročné grafické prvky pričom hlavnú časť tvorí zobrazenie mapy.

Vlastný ListView

ListView umožňuje jednoduché implementovanie zoznamu s predom definovaným zobrazením položiek. Miestnosť môže mať rôzny počet osôb, ktoré v tejto miestnosti pracujú, takže tvoria zoznam pre každú miestnosť. Pre sprehladnenie tohto zoznamu som implementoval vlastné zobrazenie položky zoznamu a pomocou triedy `PersonAdapter` som tieto položky zobrazoval.

Položka zoznamu je implementovaná za pomoci ConstraintLayout. Pomocou tohto layoutu môžem pridať komponenty na určité miesto a pričom môžem definovať vzťah medzi týmito komponentami a to hlavne odstupy medzi komponentami v pixeloch čo zabezpečí stále rovnaké umiestnenie aj za rôznych veľkostiach obrazovky. Pre vytvorenie avatara som využil *ImageView*, do ktorého je pridaný obrázok avatara. Zobrazenie mena je zabezpečené komponentou *TextView*. Pre oddelenie jednotlivých prvkov zoznamu slúži čiara implementovaná pomocou lineárneho layoutu s výškou 4dp (Density-independent Pixels) a vyplnením

[1] ConstraintLayout – https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html

pozadia farbou. Pole s menom a oddeľovacia čiara sú vždy vzdialené od avatara vo vzdialenosti 16dp a tak tvoria vzťah v ConstraintLayout.

Obrázek 6.1: Položka zoznamu pre zobrazenie osoby príslušiacej miestnosti. Položka je tvorená avatarom a menom osoby.

Navigačné posuvné okno

Interakcia, ktorá spočíva v zobrazení informácií o miestnosti, ktorú užívateľ vyhľadal alebo vybral pomocou kliknutia na mapu je implementovaná pomocou triedy *AndroidSlidingUpPanel* nachádzajúcej sa v knižnici[2]. Uvedený grafický prvok umožňuje zobraziť časť Panelu na spodu obrazovky pričom je ďalšia časť panelu schovaná pod obrazovkou. Uschovanú časť je možné zobraziť potiahnutím nahor alebo kliknutim na zobrazenú časť panelu. Prípadné spätné uschovanie je vykonané pomocou potiahnutia nadol alebo kliknutia na časť panelu.

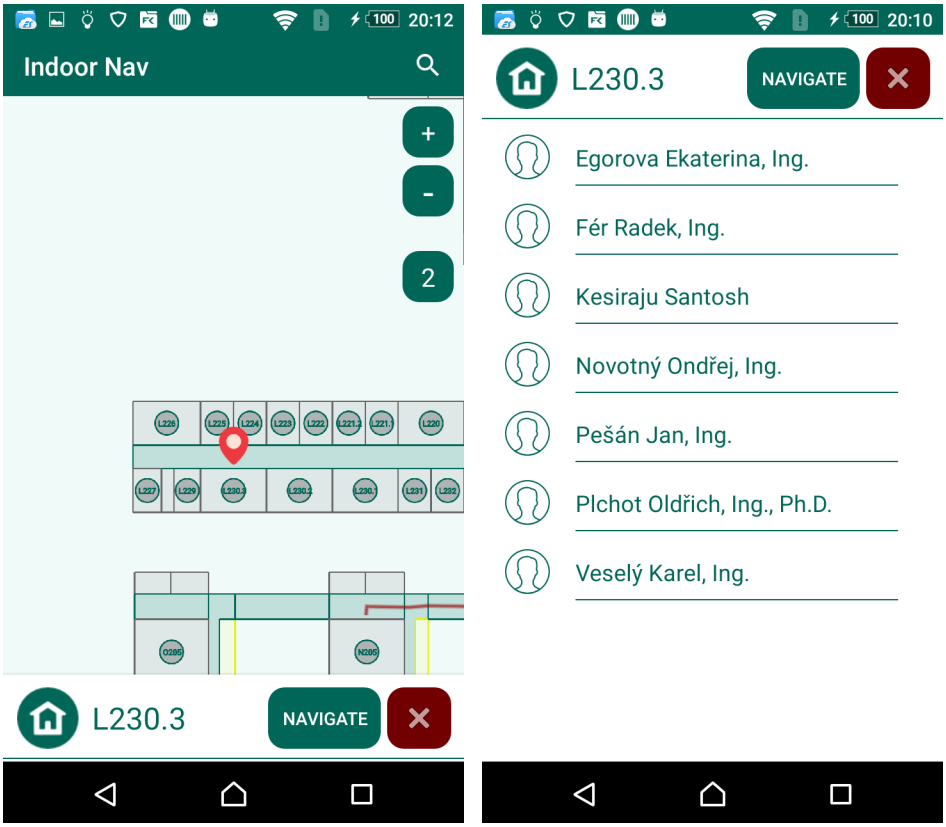
Prvá časť panelu je tvorená pomocou *LinearLayout*, ktorý má definovanú výšku 68dp. Do layoutu som vložil *ImageView*, ktorý slúži na zobrazenie obrázku s motívom budovy. Ďalej je pridaný *TextView* informujúci o názve miestnosti, ktorá bola užívateľom vybraná. Pre interakciu slúžia dve tlačidlá s textom navigate a ktoré je použité na začatie výpočtu najkratšej cesty z aktuálnej pozície. Červené tlačidlo s krížikom sa využíva pre skrytie celého tohto panelu. Druhú časť panelu tvorí Listview, ktorý obsahuje mnou implementované položky, ktorých detail je popísaný o pár riadkov vyššie. Na obrázku 6.2 sú zobrazené dve časti SlidingPanelu.

Obrazovka s mapou

Hlavnou časťou grafického rozhrania je zobrazenie mapy s príslušnými miestnosťami. Hlavnou podmienkou pre zobrazenie je správny formát vstupného súboru definujúci geometriu miestností, ktorého formát je navrhnutý v kapitole 4. Pomocou tohto súboru môže byť vykreslená mapa. Vykreslenie prebieha vo vlastnej triede `MyImageView`, ktorá dedí vlastnosti `Imageview`. Vo vytvorenej triede je implementovaná metóda `onDraw`, ktorá má ako parameter `Canvas`[3], na ktoré je možné vykreslovať rôzne vlastné grafické prvky ako body, kruhy alebo bitmapy.

Pre vykreslenie jednotlivých miestností je zavedaná trieda `Room` implementujúca vykreslenie polygónu pomocou grafickej komponenty `Path`, ktorá umožňuje vykresliť cestu z bodov. `Path` poskytuje dve metódy `MoveTo` a `LineTo`. Prvá spomenutá metóda značí začiatok cesty a druhá metóda pridá cestu z predošleho bodu do aktuálneho bodu. Týmto spôsobom je možné uzavrieť vykreslovaný objekt a nastavením vlastnosti `FillType` je zapnuté vyplnenie vnútorného priestoru objektu `Path`. Farba výplne je vybraná v závislosti od typu miestnosti. V triede `Room` je taktiež implementované vykreslenie identifikácie miestnosti na mape, pričom identifikácia je zobrazená len pre tie miestnosti, ktoré majú určitú veľkosť

[2] SlidingPanel – https://github.com/umano/AndroidSlidingUpPanel
[3] Canvas – https://developer.android.com/reference/android/graphics/Canvas.html



(a) Vykreslená len časť sliding panelu s informáciou o vybranej miestnosti a možnosti navigovania k tomuto cieľu.

(b) Úplne rozvynutý Sliding panel, kde je zobrazený zoznam osôb sídliacich v miestnosti.

Obrázek 6.2: Zobrazenie Sliding panelu pre zobrazenie informácií o vyhľadanej miestnosti.

alebo tvar (štvoruholník). Identifikačná značka je vykreslená pomocou metódy `drawCircle`, ktorá vykreslí kruh v strede miestnosti a metódou `drawText` je vykreslený názov miestnosti. V prípade schodiska nahrádza metódu `drawText` metóda `drawBitmap` pre vykreslenie obrázku schodov v kruhu.

Práca s mapou

Interakcia so zobrazením mapy je dôležitým prvkom aplikácie, ktorá sa zaoberá navigovaním, preto aplikácia nesmie zaostávať. Po vytvorení zobrazenia mapy som pridal takzvaný *OnTouchListener*[4] čo je rozhranie poskytnuté pre triedu *View* a jej podtriedy ako napríklad mnou použitý *ImageView*. Rozhranie umožňuje implementovať detekciu dotykových udalosti na obrazovke, tieto udalosti sa rozdeľujú podľa typov ako napríklad `ACTION_DOWN`, ktorý signalizuje dotyk s obrazovkou, pohyb je signalizovaný pomocou udalosti `ACTION_MOVE` a ďalším príkladom udalosti využitým v implemetácií aplikácie je `ACTION_UP`, ktorý určuje uvolnenie prsta z obrazovky.

[4] OnTouchListener – https://developer.android.com/reference/android/view/View.OnTouchListener.html

Zväčšovanie a približovanie som implementoval pomocou *ScaleDetectoru*[5], ktorý umožňuje meniť mierku zobrazenia *Canvasu* na základe gesta, ktoré bolo vykonané na obrazovke. Taktiež pre priblíženie a oddialenie je možno využiť tlačidlá, ktoré sú neustále zobrazené na obrazovke so znakmi plus k priblíženiu respektíve mínus k oddialeniu.

Ďalšou interaktívnou funkciou, ktorú užívateľ môže využiť je voľba cieľa pre navigovanie pomocou výberu miestnosti na obrazovke. Pri využití *OnTouchListeneru* pre detekovanie gést je nutné implementovať detekciu kliknutia na základe doby pohybu na obrazovke. Tento spôsob získania udalosti kliknutia následne nenarúša priebeh v prípadnom vykonaní gesta pohybu mapy. Čas, ktorý oddeľuje kliknutie od ťahania mapy som stanovil na 100ms. K tejto hodnote som prišiel k neustálemu skúšaniu zmeny danej hodnoty až sa napokon vykryštalizovala uvedená hodnota.

Implementoval som aj automatické presunutie na lokalizovanú polohu. To znamená, že po prehľadávaní mapy užívateľom sa spustí odpočítadlo, po ktorom sa aplikačné okno nastaví na užívateľskú polohu. Myslim, že táto funkcionalita môže byť prospešná, ak sa užívateľ stratil v mape a nevie si sám nájsť polohu, pričom podobnú funkcionalitu vykonáva tlačidlo v pravom dolnom rohu pre lokalizovanie.

6.2 Lokalizovanie

Implementácia lokalizačného algoritmu využívajúci metódu WIFI fingerprinting je zaobstaraná knižnicou [6]. Knižnica spočíta vzdialenosti medzi jednotlivými fingerprintami a vráti ten fingerprint, ktorý je najbližší k súčasnej skenovanej vzorke. Po získaní najbližšieho fingerprintu je vykreslený PinPoint, ktorého funkcionalita je popísaná nižšie.

Zobrazenie PinPoint

rá Lokalizovanie by sa neobišlo bez zobrazenia bodu na mape, v ktorom sa užívateľské zariadenie nachádza. K zobrazeniu som klasicky pre navigáciu zvolil PinPoint, čo je obrázok v tvare trojuholníka so zaguľatelými dvoma vrcholmi. Pre navigáciu je typické, že sa tento PinPoint hýbe a zobrazuje tak smer pohľadu užívateľa. Implementovanie takejto funkcionality je možné pomocou inerciálnych jednotiek a to hlavne pomocou akcelerometra a magnetometra, ktoré sú zabudované už v každom mobilnom zariadení. Akcelerometer využívam pre detekovanie pohybu zariadenia voči prostrediu a magnetometer pre určenie smeru.

Na to, aby bolo možné pristúpiť k senzorom je v Android SDK poskytnutý `SensorManager`[7]. Inštancia tejto triedy sa získava pomocou systémovej žiadosti o prístup k senzorom metódou `getSystemService(SENSOR_SERVICE)`. Ako som už spomenul využijem akcelerometer (`TYPE_ACCELEROMETER`) a magnetometer (`TYPE_MAGNETIC_FIELD`). Tieto senzory sú ďalej využité pre výpočet rotačnej matice a následne pomocou rotačnej matice prebieha výpočet orientácie zariadenia pomocou vstavanej funkcie `getOrientation`, ktorá vracia pole troch prvkov udávajúcich azimuth, stúpanie a otočenie. Vyhľadávame smer na základe svetovej orientácie, takže využijem len prvú položku a tou je azimuth. Azimuth definuje uhol medzi

[5] ScaleDetector – https://developer.android.com/reference/android/view/ScaleGestureDetector.html

[6] Knižnica lokalizovanie – https://github.com/TeroM/indoor-position-tracker

[7] SensorManager – https://developer.android.com/reference/android/hardware/SensorManager.html

y–súradnicou a Severným pólom, ktorý je udávaný v Radiánoch. Následne je uhol prepočítaný na stupne a využitý v animácií, ktorá vykreslenú bitmapu rotuje na základe získaného uhlu. Pričom zariadenia s veľkosťou uhlu 0 smeruje na juh, s veľkosťou π smeruje na sever, východ je určený ako $\frac{\pi}{2}$ a v prípade západu $-\frac{\pi}{2}$. Avšak pri budovách s veľkou kovovou konštrukciou nastáva deviácia kompasu a teda rušenie presnosti kompasu na základe zmeny magnetického poľa, kvôli železnej konštrukcií.

6.3 Navigácia a vyhľadávanie

Navigácia je reprezentovaná grafom, ktorý definuje možné cesty v mape.Cesta sa v grafe vyhľadá pomocou Dijkstrovho algoritmu, ktorý je popísaný v kapitole 3. Počiatočný bod cesty je definovaný ako najbližší vrchol grafu od pozície užívateľa na mape a koncový bod cesty je vyhľadaný užívateľom, či už to je kliknutím na mapu alebo vyhľadaním v zozname. Takto vyhľadaná cesta je následne vykreslena pomocou vykresĺovacích funkcií `drawPath`, pričom je cesta vykreslená podľa poschodí. Pri prepnutí poschodia sa vykresli iná časť cesty, ktorá prináleži k vybranému poschodiu.

Vyhľadávanie

Vyhľadávania som implementoval pomocou knižnice[8]. Táto knižnica obsahuje rôzne možnosti vyhľadávania či už klasické pomocou písania textu alebo dokonca aj vyhľadávanie pomocou hlasu. Pri vyhľadávaní pomocou písania textu do textového poľa sa zobrazuje zoznam tých položiek, ktoré vyhovujú vyhľadávanému reťazcu. Aby bolo vyhľadávanie možné vzniká potreba prideliť k `MaterialSearchView` textové návrhy, cez ktoré bude vyhľadávania vykonané. V navigáciách vo vnútorných priestoroch sa zvyčajne vyhľadávajú osoby alebo kancelárie. Tieto kancelárie a osoby sú doplnené zo zoznamu ľudí definovaných v XML súboroch (kapitola 4) a uložím si ich do poľa reťazcov. Vytvorené pole návrhov nastavím do inštancie triedy `MaterialSearchView` pomocou metódy `setSuggestions`.

Po nastavení návrhov a následnom vyhľadaní užívateľom osoby alebo miestnosti je vytvorený dotaz do databáze a to taký, že sa najprv dotazuje do tabuľky miestností, či existuje nejaká zhoda s vybraným textovým reťazcom, ak áno tak sa vráti vyhľadaná miestnosti. Ak taká miestnosť neexistuje prejde sa na druhý dotaz, ktorým je prehľadaná tabuľka osôb pričom je v tomto dotaze vytvorené spojenie dvoch tabuliek osoby a miestnosti, aby bolo možné vrátiť miestnosť, kde vyhľadávaná osoba sídli. V tejto miestnosti je definovaný vrchol grafu pre vyhľadanie cesty v grafe a je taktiež vyznačený na príslušnom poschodí.

6.4 Načítanie dát

Vstupné XML súbory definujúce reprezentáciu budovy, databázu fingerprintov, graf a v neposlednom rade zoznam osôb príslušných k miestnostiam. Tieto súbory sú rozobrané a postupne sú pridelené do triednych reprezentácií pre jednotlivé súbory. Parsovanie prebieha pomocou `XmlPullParser`[9]. čas tohto procesu vzniká aj databáza, ktorá sa plní hodnotami zo súborov.

[8] MaterialSearchView – https://github.com/MiguelCatalan/MaterialSearchView
[9] XmlPullParser – https://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html

Databáza je implementovaná pomocou knižnice *SQLite*[10]. SQLite je najviac rozšírenou databázou na svete, pri ktorej nemusí bežať oddelený serverový proces. Pri využití v prostredí Androidu poskytuje internú databázu pre aplikáciu, ktorá je dostupná len pre dané zariadenie.

Aby bolo možné využiť SQLite v prostredí Androidu, musím implementovať `SQLiteOpenHelper`, pomocou ktorého môžem vytvoriť tabuľky poprípade sa dotazovať pomocou príkazov `SELECT`. Vytvorenie tabulky je zahrnuté v metóde `onCreate`, ktorá sa volá pri vytvorení databáze a obsahuje všetky príkazy `CREATE` vytvárajúce tabuľky.

Tabuľky som z počiatku plnil po jednej položke, čo sa podpísalo pod pomalým zahájaním aplikácie, ktorej spustenie trvalo aj niekoľko sekúnd pri veľkom počte miestností. Rýchlosť spustenia aplikácie môže užívateľa jednoznačne odradiť od použitia, preto som sa to snažil urýchliť. Urýchlenie sa mi podarilo vytvoriť pomocou transakcií. Transakcie tvoria skupinu príkazov, ktoré prevedú databázu z jednoho konzistentného stavu do druhého. Transakcie urýchlili proces načítania dát veľkým spôsobom a aplikácia sa tak spúšťa hned po sekunde–dvoch.

6.5 Testovanie a experimentovanie

Na mobilnej aplikácií som testoval samotný postup a to možnosť navigovania pomocou vytvoreného grafu z pomocného nástroja. Na testovanie boli z počiatku využívané emulátory, ktoré sú priložené v balíku SDK vo vývojovom prostredí Android Studio. Emulátor poskytuje možnosti zobrazenia vyvíjanej aplikácie, pričom výhodou emulátorov je ich prispôsobenie podľa vlastnej predstavy. Android studio ponúka viacero zariadení, pomocou ktorých možno otestovať zobrazenie aplikácie na rôznych veľkostiach zariadení poprípade zariadení, ktoré bežia na iných verziach operačného systému Android. Emulátor som využíval pri testovaní funkčnosti grafického rozhrania a pri testovaní funkčnosti ukazovania smetu pohľadu užívateľa vzhľadom k budove. No nevýhodou emulátorov je vhľadom k výkonu počítača veľmi pomalá a preto som prešiel na reálne zariadenia a to konkrétne Sony Z3 Compact s Androidom 6.0.1 a Lenovo P70–A s Androidom 5.1. Mobilné zariadenie mi prinieslo jednoduchšiu prenositeľnosť a to hlavne pri testovaní lokalizačnej metódy v konkrétnej budove.

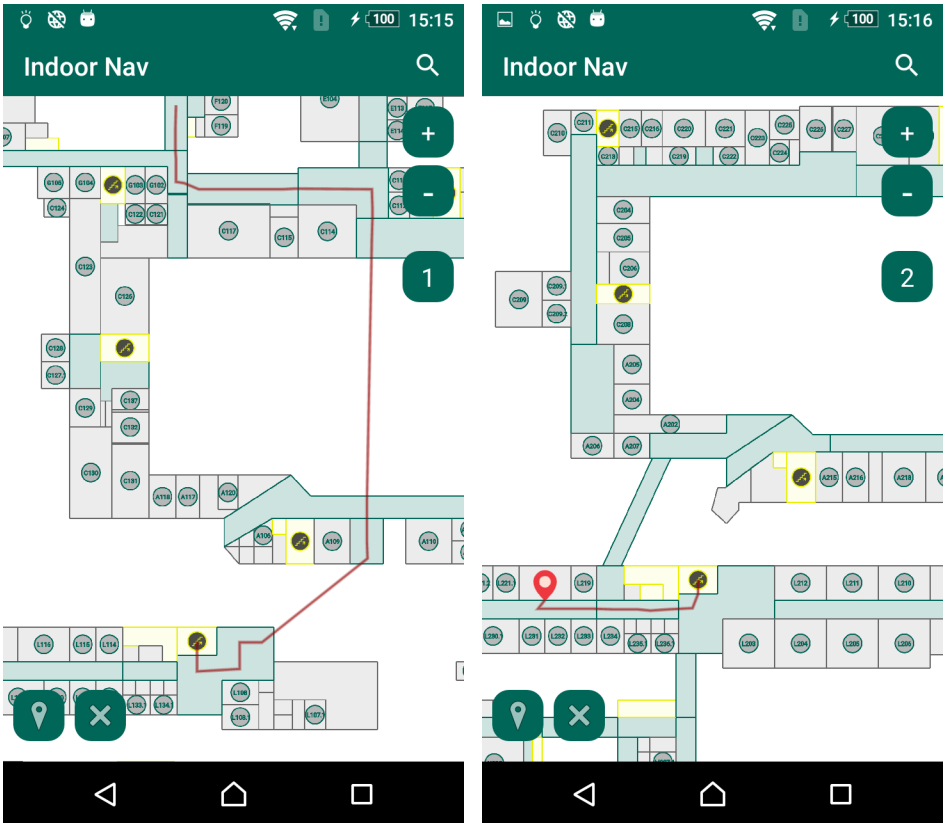
Spojitosť navigačného grafu

Testovanie spojitosti grafu som zobral najprv z tej jednoduchšej stránky a to konkrétne testovaním spojitosti len na poschodí a to tak, že som si určil pevne daný počiatočný bod a hľadal som cesty do miestností, ktoré sa na danom poschodí nachádzajú. Spojitosť grafu vyšla vo výslednom testovaní úspešne a cesty na jednom poschodí sú jednoducho navigovateľné. Myslím, že dobré výsledky sú zapríčinené najmä prispôsobením a zobrazením grafu pri tvorbe podľa užívateľa a on tak vidí jeho spojitosť, poprípade môže doplniť chýbajúce spojenia.

Avšak problématickou časťou generovania sú prepojenia poschodí, kde užívateľ už nie je zapojený do procesu spájania a prepojenia vytvára automaticky program. Pri testovaní som postupoval obdobne ako v predošlých testoch. Prvotným testovaním sa potvrdilo, že grafy sú spojené dobre medzi poschodím. Avšak pri testovaní spojitosti medzi pochodiami sa našli výnimky, kedy sa zdalo, že graf nebol prepojený. Bližším skúmaním tohto problému som došiel k tomu, že k problému došlo pri spájaní vrcholov schodísk s vrcholmi chodby,

[10] SQLite – https://www.sqlite.org/index.html

kedy sa nevytvorilo spojenie s priliehajúcou chodbou a tým pádom nemohla byť vypočítaná najkratšia cesta do iných poschodí. Problém som vyriešil a dokázal som získať cesty aj medzi poschodiami, pričom ukážka spojenia je na obrázku č. 6.3, kde je zobrazená cesta z prvého poschodia do druhého.



(a) Počiatočný bod v okolí rektorátu fakulty na prvom poschodí a vyobrazené prepojenie poschodí cez schodisko

(b) Cieľovým bodom je miestnosť L220 na druhom poschodí, taktiež je vyobrazený spoj poschodí cez schodisko.

Obrázek 6.3: Experiment so spojitosťou grafu.

Presnosť trajektórie lokalizovania

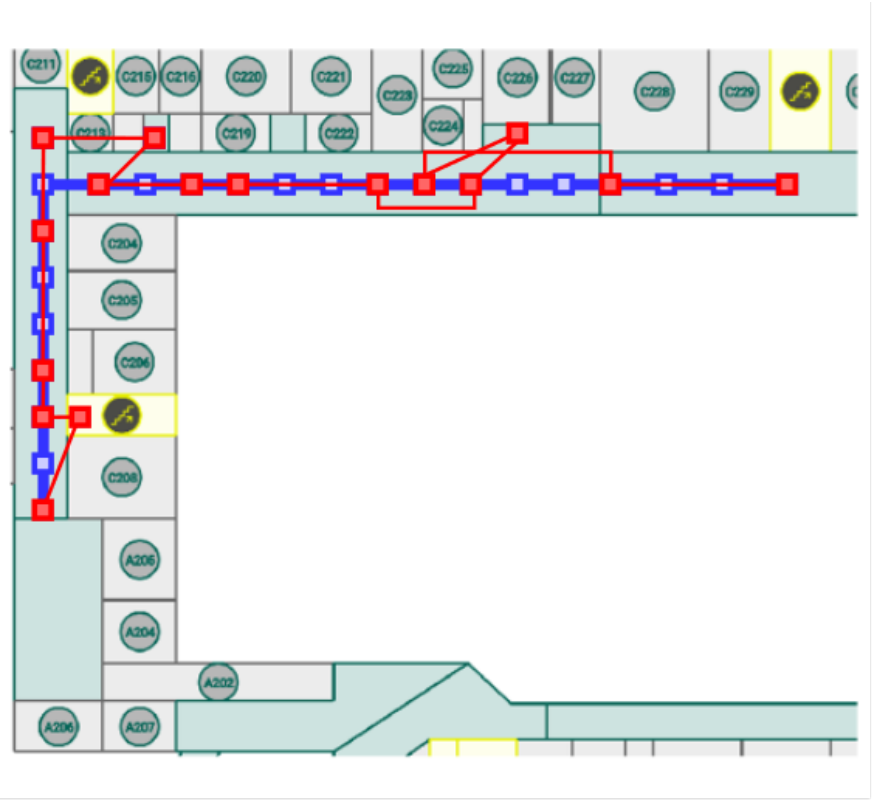
Najdôležitejším ukazateľom každého navigačného systému je presnosť určenia polohy užívateľa. Testovanie som navrhol takým spôsobom, že si určím predpokladanú trasu a tú budem nasledovať s mobilnou aplikáciou, kde budem zachytávať jednotlivé body, cez ktoré som prešiel a tak budem môcť vidieť rozdiely v trasách. Vďaka tomuto testovaniu, zistím akú presnú trajektóriu pohybu metóda WIFI fingerprinting umožní zaznamenať.

Trasa chôdze, na obrázku 6.4 zobrazená modrou farbou, bola zaznamenaná na základe databáze fingerprintov, pričom obraz bol doupravený kvôli lepšej prehľadnosti tejto cesty. Tento experiment bol vykonaný dva týždne po vykonaní zberu dát do databáze fingerprintov, takže sa dá predpokladať, že lokalizovanie nebude najpresnejšie vďaka ovplyvneniu síl signálov od jednotlivých access pointov časom, toto zistenie je prezentované v sekcií č. 5.4.

Na obrázku 6.4 je možno vidieť trasu, ktorá bola získaná na základe lokalizácia vyznačená červenou farbou. Z vytvorenej cesty je vidieť, že lokalizovanie sa snaží kopírovať

prejdenú trasu, ale je vidieť, že sú tam určité výchylky, ktoré môžu byť spôsobené či už časom alebo aj počtom access pointov v tomto okolí. Pri zbere týchto dát sa objavovali polohy rovnaké a teda podľa lokalizácie som neuskutočnil žiaden pohyb, ale opak bol pravdou. Ďalej je vidieť, že sa lokalizačná metóda vracala späť a neišla so smerom pohybu. Vo výsledku však bolo možné určiť aspoň približnú polohu na základe lokalizácie, ktorá bola v najhoršom prípade vzdialená od reality nie toľko, aby bol užívateľ stratený v mape.

Počas testovania trajektórie v iných lokalitách fakulty som narazil na nie až tak presné určenia poschodia, kedy sa problém najviac vyskytoval v okolí schodísk, ktoré prepájajú jednotlivé poschodia. Lokalizačná metóda v týchto priestoroch viacero krát prepla poschodia aj bez toho, že som po schodoch nešiel. Väčšinou však po ďalšom lokalizačnom kroku sa poloha poschodia znovu vrátila do normálnych kolají a prepla sa obrazovka na správne poschodie.



Obrázek 6.4: Vyobrazenie experimentu s lokalizovaním pomocou trasy, kde červená trasa je získaná z lokalizačnej metódy a modrou farbou je vyznačená trasa chôdze.

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť postup, ktorým sa zjednoduší nasadzovanie navigačného a lokalizačného systému do vnútorných priestorov budov. Pričom bolo cieľom aj vytvoriť mobilnú aplikáciu, ktorá tento postup realizuje a demonštruje. Po naštudovaní potrebných informácií o lokalizačných metódach a možnostiach tvorby navigácie som tento postup navrhol a následne aj implementoval. Na riešenie som využíval internetové zdroje, ktoré mi poskytli veľké množstvo užitočných informácií.

Samotné prvé dve kapitoly sa venuju z prvej časti potrebnej teórií a prehľadu metód, ktoré môžu byť využité na lokalizovanie vo vnútorných priestoroch. Po týchto nasleduje kapitola venujúca sa návrhu riešenia. Ďalej sú uvedené kapitoly venujúce sa implementačným detailom navrhnutého postupu a mobilnej aplikácií. Nakoniec je uvedené testovanie vykonané na mobilnom zariadení.

Postup zjednodušenia tvorby navigačného systému je obsiahnutý v pomocnom nástroji. V pomocnom nástroji je zobrazená mapa, ktorá je pridelená XML súborom pomocou užívateľa. Keďže som využíl pre lokalizáciu metódu wifi fingerprinting, implementoval som do tohto nástroja časť, ktorá po kliknutí na miesto na mape oskenuje všetky access pointy v okolí a uloží ich do databáze fingerprintov. Ďalej je zahrnuté poloautomatické generovanie grafov na základe geometrie miestností a pomocou užívateľa dodefinovať navigačný graf. Výstupom sú súbory defunujúce databázu fingerprintov, graf a miestnosti (vstupné súbory).

Následne po implementácií pomocného nástroja nasledovala implementačná časť, ktorá realizuje navrhnutý postup na mobilnom zariadení. Aplikácia bola implementovaná pomocou wireframu, ale vzhľadom jeho jednoduchosti som pridal iné užitočné komponenty ako napríklad slidingPanel. Po manuálnom vložení súborov do aplikácie sa súbory spracujú a implementujú jednotlivé časti. V mobilnej aplikácií je možné vyhľadávať pomocou mena miestnosti alebo podľa mena osoby, ktorá v miestnosti sídli. Lokalizovanie prebieha pomocou už spomenutej metódy wifi fingerprinting, kedy prebieha porovnanie aktuálnych fingerprintov s databázou.

Postup tvorby navigačného systému bol otestovaný na mobilnej aplikácií, ktorá implementovala budovu Fakulty informačných technológií Vysokého Učení technického v Brně. Testovanie bolo zamerané na prepojenie grafu a to najmä prepojenie medzi poschodiami. Ďalej bol skúmaný dopad času na zbierané fingerprinty a nakoniec bol vykonaný test, ktorý bol zameraný na presnosť lokalizovanej trasy.

Súčasné riešenie implementuje zber fingerprintov na desktopovej aplikácií, čo by bolo dobré zaintegrovať do mobilnej aplikácie. Týmto by sa odstránila potreba notebooku, ale aj nástroja *iwlist*, takže tým pádom by bolo skenovanie aj rýchlejšie. Do mobilnej aplikácie

by som taktiež zakomponoval generovanie grafu s tým rozdielom, že generovanie by bolo plne automatické a využívalo by skeletonizáciu chodieb, kedy by sa prepojenie dverí spojilo s najbližším bodom získaného skeletonu. Pre následné jednoduchšie zdielanie vytvorených máp by bolo dobré vytvoriť serverovú časť, kde by sa takto vytvorené mapy dali uložiť do databázy a iný užívateľ by si ich následne mohol stiahnuť do mobilnej aplikácie.

Literatura

[1] Bai, Y. B.; Norman, R.; Zhao, Y.; aj.: *A New Algorithm for Improving the Tracking and Positioning of Cell of Origin*. 2015, [Online; navštívené 15.10.2017].
URL http://ieeexplore.ieee.org.ezproxy.lib.vutbr.cz/stamp/stamp.jsp?tp=&arnumber=7352267

[2] de Berg, M.; Cheong, O.; van Kreveld, M.; aj.: *Computational Geometry Algorithms and Applications*. Springer, 2008, ISBN 978-3-540-77973-5.

[3] Dijkstra, E. W.: *A Note on Two Problems in Connexion with Graphs*. [Online; navštívené 15.10.2017].
URL http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=08FC7F1E0971AA72149227C4526CB474?doi=10.1.1.165.7577&rep=rep1&type=pdf

[4] Gustafsson, F.; Gunnarsson, F.: *Positioning using Time-difference of arrical measurements*. Department of Electrical Engineering Linkoping University, 2013, [Online; navštívené 12.10.2017].
URL https://pdfs.semanticscholar.org/4090/577dc06b844a73c3855df9cc751107acd9c7.pdf

[5] Henniges, R.: *Current approches of Wifi Positioning*. 2012, [Online; navštívené 10.10.2017].
URL https://www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1112/snet-project/wifi-positioning_henniges.pdf

[6] Kolář, J.: *Teoretická informatika*. Česká informatická společnost, 2000, ISBN 80-900853-8-5.

[7] Kršek, P.: *Základy počítačové grafiky : IZG*. Brno : Fakulta informačních technologií, 2008.

[8] Nathaniel Bowditch, L.: *The Amarican practical navigator*. National imagery and mapping agency, 1995, [Online].
URL https://web.archive.org/web/20060313165051/http://www.irbs.com/bowditch/pdf/chapt07.pdf

[9] Navarro, E.; Peuker, B.; Quan, M.: *1 Wi-Fi Localization Using RSSI Fingerprinting*. California Polytechnic State University, USA, 2011, [Online].
URL http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1007&context=cpesp

[10] Piciarelli, C.: *Visual Indoor Localization in Known Environments* . IEEE, Říjen 2016, [Online].
URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7519056

[11] Pratama, A. R.; Widyawan; Hidayat, R.: *Smartphone-based Pedestrian Dead Reckoning as an Indoor Positioning System* . Information Technology and Electrical Engineering Department, Gadjah Mada University , Indonézia, Září 2012, [Online].
URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6339316

[12] Ravindra, S.; Jagadeesha, N. S.: *Time of arrival based localization in wireless sensor networks: A linear approach*. 2013, [Online; navštívené 10.10.2017].
URL http://aircconline.com/sipij/V4N4/4413sipij02.pdf

[13] Rong, P.; Sichitiu, M. L.: *Angle of Arrival Localization for Wireless Seonsor Networks*. IEEE, 2016, [Online; navštívené 12.10.2017].
URL http://aircconline.com/sipij/V4N4/4413sipij02.pdf

[14] Turzík, D.; Pavlíková, P.: *Diskrétni matematika*. VŠCHT Praha, 2007, ISBN 978-80-7080-667-8.

[15] Xiong, B.; Zheng, Z.: *Graph theory*. Shanghai : East China Normal University Press ; Singapore : World Scientific Publishing, 2010, ISBN 978-981-4271-12-7.

[16] Yang, L.; Worboys, M.: *Generation of navigation graphs for indoor space*. School of Computing and Information Science, University of Maine, USA, 2014, [Online; navštívené 20.12.2017].
URL http://gala.gre.ac.uk/13719/1/
13719_WORBOYS_Generation_of_navigation_AAM_(2015).pdf

Příloha A

Obsah priloženého DVD

Štruktúra priloženého DVD

- **MobileApp**
  - Súbory so zdrojovými kódmi aplikácie
  - Inštalačný súbor aplikácie – APK
- **Tool**
  - Súbory so zdrojovými kódmi nástroja
  - Inštalačný súbor nástroja – JAR
- **Poster** – Prezentačný plagát
- **Video** – Prezentačný video
- **Latex** – Zdrojový kód textovej časti
- **Pdf** – Elektronická forma textu práce
- **manual.pdf** – Návod na inštaláciu aplikácie a nástroja